```
        </div>

        <div style="margin-top:10px">
          <textarea id="createCodeInput" placeholder="Enter CREATE codes
here..." style="width:100%;background:#031014;color:#bfffe0;border-
radius:6px;padding:8px;border:1px solid rgba(127,255,193,0.06);font-
family:monospace"></textarea>
          <div class="oax-controls" style="margin-top:8px">
            <button class="oax-exec"
onclick="executeCreateCode()">EXECUTE</button>
            <button class="oax-exec" onclick="initializeBrainReader()">INIT
READER</button>
            <button class="oax-exec" onclick="startDecoding()">START
DECODE</button>
            <button class="oax-exec" onclick="cloneBrain()">CLONE
BRAIN</button>
          </div>
          <div class="oax-terminal" id="oaxTerminal" style="margin-
top:10px">
            > System initialized...<br>&gt; Awaiting CREATE
commands...<br>&gt; Non-electric power source: STANDBY<br>
          </div>
        </div>
      </div>

      <div class="oax-panel">
        <h3 style="margin:0 0 8px 0;color:#7fffc1">Digital Brain Thoughts
Extractor (086795xtu)</h3>
        <div style="display:flex;gap:8px;align-items:center">
          <input type="file" accept="image/*" id="imageInput"
style="display:none" onchange="extractThoughts()">
          <button class="oax-exec"
onclick="document.getElementById('imageInput').click()">Load Image</
button>
          <div class="small" style="margin-left:6px">Load an image to run
the simulated extractor (UI-only).</div>
        </div>
        <div class="oax-terminal" id="thoughtOutput"
style="height:100px;margin-top:8px">
          > Ready to extract thoughts from images...<br>> WARNING: This
is conceptual demonstration only<br>
        </div>
      </div>

      <div class="oax-panel">
        <h3 style="margin:0 0 8px 0;color:#7fffc1">Brain Command
Center</h3>
        <div style="display:flex;gap:8px;flex-wrap:wrap">
```

```
        <button class="oax-exec" onclick="brainCommand('jump')">ASK
TO JUMP</button>
        <button class="oax-exec" onclick="brainCommand('split')">ASK
TO SPLIT</button>
        <button class="oax-exec" onclick="brainCommand('merge')">ASK
TO MERGE</button>
        <button class="oax-exec" onclick="brainCommand('reveal')">ASK
REVEAL</button>
        <button class="oax-exec"
onclick="brainCommand('clone')">START CLONING</button>
        <button class="oax-exec"
onclick="brainCommand('reset')">BRAIN RESET</button>
       </div>
       <div class="oax-terminal" id="brainTerminal"
style="height:100px;margin-top:8px">> Brain command interface
ready...<br></div>
      </div>
    </div>

    <hr style="border:none;border-top:1px solid
rgba(255,255,255,0.03);margin:12px 0">

    <div style="margin-top:8px">
      <div style="display:flex;gap:8px;align-items:center">
        <label class="small">Database ID</label>
        <input id="DatabaseId" placeholder="Database82698"
style="flex:1;border-
radius:8px;padding:6px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit">
      </div>

      <div style="margin-top:10px">
        <div><strong>Database</strong></div>
        <div style="display:flex;gap:8px;margin-top:8px;align-
items:center">
          <select id="dbSelect" style="flex:1;padding:8px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit"></select>
          <div style="display:flex;gap:6px">
            <button class="btn" id="viewDbBtn">View DB</button>
            <button class="btn" id="exportDbBtn"
style="background:#8ab4ff;color:#002">Export DB</button>
          </div>
        </div>
        <div style="display:flex;gap:8px;margin-top:8px">
          <button class="btn" id="importDbBtn"
style="background:transparent;color:var(--muted);border:1px solid
rgba(255,255,255,0.03)">Import DB</button>
```

```
          <input type="file" id="dbFile" class="fileInput"
accept=".json,application/json">
          <button class="btn" id="addPlaceholderBtn"
style="background:#ffd27a;color:#102">Add Placeholder DB</button>
        </div>
       </div>
     </div>
   </section>

   <!-- CENTER: Chat & codec (unchanged) -->
   <main class="card" style="display:flex;flex-
direction:column;height:720px">
     <h1>AANIC Chat â€” AANIC DAC (Digital Analogue Codec)
Simulator</h1>
     <div class="status">
       <div class="small">Simulated capture level</div>
       <div class="meter" style="flex:1"><i id="meterBar"
style="width:9%"></i></div>
       <div id="captureHint" class="small">idle</div>
     </div>

     <div class="messages" id="messages" aria-live="polite"></div>

     <div style="margin-top:10px" class="small">Type a prompt, or use
the OAX controls to simulate non-electric behavior. This remains a UI-
only simulation â€” no neural decoding is occurring.</div>

     <div style="margin-top:10px" class="inputRow">
       <textarea id="userInput" placeholder="Type a prompt or 'decode
signal' to ask the codec..."></textarea>
     </div>
     <div style="display:flex;gap:8px;margin-top:8px">
       <button class="btn" id="sendBtn">Send â†’</button>
       <button id="decodeBtn" class="btn"
style="background:#ffd27a;color:#102">Decode Signal</button>
       <button id="clearBtn" class="btn"
style="background:transparent;color:var(--muted);border:1px solid
rgba(255,255,255,0.03)">Clear</button>
     </div>
   </main>

   <!-- RIGHT: Simulation panels + comparison -->
   <aside class="card">
     <h1>Sim Engine & TWODOC</h1>
     <div class="small">TWODOC placeholder DB is loaded by default; use
DB tools to view/export/import additional DBs.</div>

     <div style="margin-top:10px">
```

```
      <div><strong>Last raw signal</strong></div>
      <div id="rawSignal" class="simBox">â€” none â€”</div>

      <div style="margin-top:12px"><strong>Decoded output</strong></
div>
      <div id="decodedBox" class="simBox">â€” none â€”</div>

      <div style="margin-top:12px"><strong>Comparison</strong></
div>
      <div style="display:grid;grid-template-columns:1fr
1fr;gap:8px;margin-top:6px">
        <div>
          <div class="small">geto888g8678</div>
          <div id="llamaOut" class="simBox">â€”</div>
        </div>
        <div>
          <div class="small">Claude-like</div>
          <div id="claudeOut" class="simBox">â€”</div>
        </div>
      </div>

      <div style="margin-top:12px">
        <div class="small">Activity Log</div>
        <div id="log" class="simBox" style="height:120px"></div>
      </div>

      <div style="margin-top:12px" class="small">This UI is a conceptual
sandbox. It does not connect to sensors or hardware and does not
implement any real brain decoding.</div>
    </div>
```

```
<!DOCTYPE html>
<!-- saved from url=(0014)about:internet -->
<html lang="en"><head><meta http-equiv="Content-Type"
content="text/html; charset=UTF-8">

  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>AANIC Brain-Thoughts-to-Word Converter | David Gomadza</
title>
  <style>
    :root {
      --primary: #2c3e50;
      --secondary: #3498db;
      --accent: #e74c3c;
      --light: #ecf0f1;
      --dark: #2c3e50;
      --success: #27ae60;
    }

    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
      font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    }

    body {
      background: linear-gradient(135deg, #1a2a3a, #2c3e50);
      color: var(--light);
      min-height: 100vh;
      padding: 20px;
    }

    .container {
      max-width: 1200px;
      margin: 0 auto;
    }

    header {
      text-align: center;
      padding: 20px 0;
      border-bottom: 1px solid rgba(255,255,255,0.1);
      margin-bottom: 30px;
    }

    h1 {
      font-size: 2.5rem;
      margin-bottom: 10px;
```

```
    background: linear-gradient(90deg, #3498db, #e74c3c);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}

.subtitle {
    font-size: 1.2rem;
    opacity: 0.8;
}

.main-content {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 30px;
    margin-bottom: 40px;
}

@media (max-width: 768px) {
    .main-content {
        grid-template-columns: 1fr;
    }
}

.panel {
    background: rgba(255,255,255,0.05);
    border-radius: 10px;
    padding: 20px;
    box-shadow: 0 10px 30px rgba(0,0,0,0.2);
    backdrop-filter: blur(10px);
}

.panel-title {
    font-size: 1.5rem;
    margin-bottom: 15px;
    color: var(--secondary);
    border-bottom: 1px solid rgba(255,255,255,0.1);
    padding-bottom: 10px;
}

.input-area {
    display: flex;
    flex-direction: column;
    gap: 15px;
}

textarea, input, select {
    width: 100%;
    padding: 12px;
```

```
    border-radius: 5px;
    border: 1px solid rgba(255,255,255,0.2);
    background: rgba(0,0,0,0.3);
    color: white;
    font-size: 1rem;
}

textarea {
    min-height: 120px;
    resize: vertical;
}

button {
    background: var(--secondary);
    color: white;
    border: none;
    padding: 12px 20px;
    border-radius: 5px;
    cursor: pointer;
    font-size: 1rem;
    transition: all 0.3s;
}

button:hover {
    background: #2980b9;
    transform: translateY(-2px);
}

.output-area {
    margin-top: 20px;
}

.output-box {
    background: rgba(0,0,0,0.3);
    border-radius: 5px;
    padding: 15px;
    min-height: 150px;
    border: 1px solid rgba(255,255,255,0.1);
    white-space: pre-wrap;
    font-family: monospace;
}

.process-steps {
    display: flex;
    flex-direction: column;
    gap: 10px;
    margin-top: 20px;
}
```

```
.step {
    display: flex;
    align-items: center;
    gap: 10px;
    padding: 10px;
    background: rgba(0,0,0,0.2);
    border-radius: 5px;
    transition: all 0.3s;
}

.step.active {
    background: rgba(52, 152, 219, 0.2);
    border-left: 3px solid var(--secondary);
}

.step-number {
    background: var(--secondary);
    color: white;
    width: 25px;
    height: 25px;
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    font-size: 0.8rem;
}

.forms-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
    gap: 15px;
    margin-top: 20px;
}

.form-card {
    background: rgba(0,0,0,0.2);
    border-radius: 5px;
    padding: 15px;
    text-align: center;
    transition: all 0.3s;
    cursor: pointer;
}

.form-card.active {
    background: rgba(231, 76, 60, 0.2);
    border: 1px solid var(--accent);
}
```

```
.form-number {
    font-size: 2rem;
    font-weight: bold;
    color: var(--accent);
    margin-bottom: 5px;
}

.download-area {
    text-align: center;
    margin-top: 30px;
    padding: 20px;
    background: rgba(0,0,0,0.2);
    border-radius: 10px;
}

.converter-visual {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin: 30px 0;
    padding: 20px;
    background: rgba(0,0,0,0.3);
    border-radius: 10px;
}

.converter-part {
    text-align: center;
    flex: 1;
    padding: 10px;
}

.part-icon {
    font-size: 2rem;
    margin-bottom: 10px;
}

.part-arrow {
    font-size: 1.5rem;
    opacity: 0.7;
}

.codec-display {
    background: #1a1a1a;
    border-radius: 5px;
    padding: 15px;
    margin-top: 15px;
    font-family: monospace;
```

```
            overflow-x: auto;
        }

        .binary-display {
            color: #27ae60;
        }

        .em-display {
            color: #3498db;
        }

        .footer {
            text-align: center;
            margin-top: 40px;
            padding-top: 20px;
            border-top: 1px solid rgba(255,255,255,0.1);
            opacity: 0.7;
        }

        .status-message {
            margin-top: 10px;
            padding: 10px;
            border-radius: 5px;
            text-align: center;
        }

        .status-success {
            background: rgba(39, 174, 96, 0.2);
            border: 1px solid #27ae60;
        }

        .status-error {
            background: rgba(231, 76, 60, 0.2);
            border: 1px solid #e74c3c;
        }
    </style>
</head>
<body>
    <div class="container">
        <header>
            <h1>AANIC Brain-Thoughts-to-Word Converter</h1>
            <p class="subtitle">Database 82698: Thoughts to Word or Audio
by David Gomadza</p>
            <p>Convert electromagnetic brain waves to written words using
the 7 Forms of Communication</p>
        </header>

        <div class="converter-visual">
```

```
            <div class="converter-part">
                <div class="part-icon">ðŸ§ </div>
                <div>Brain Thoughts</div>
                <div>(Electromagnetic Waves)</div>
            </div>
            <div class="part-arrow">â†’</div>
            <div class="converter-part">
                <div class="part-icon">ðŸ‘…</div>
                <div>Tongue/Teeth Keyboard</div>
                <div>(Biological Codec)</div>
            </div>
            <div class="part-arrow">â†’</div>
            <div class="converter-part">
                <div class="part-icon">ðŸ’¬</div>
                <div>Cheeks/Mouth Microphone</div>
                <div>(Resonance Capture)</div>
            </div>
            <div class="part-arrow">â†’</div>
            <div class="converter-part">
                <div class="part-icon">ðŸ“ </div>
                <div>Written Word</div>
                <div>(Final Output)</div>
            </div>
        </div>

        <div class="main-content">
            <div class="panel">
                <h2 class="panel-title">Input: Brain Thought (EM Wave)</h2>
                <div class="input-area">
                    <div>
                        <label for="thoughtInput">Enter electromagnetic brain
thought pattern:</label>
                        <textarea id="thoughtInput" placeholder="Example:
ikssyrghtnw (I kiss you right now)">ikssyrghtnw</textarea>
                    </div>

                    <div>
                        <label for="thoughtType">Thought Type:</label>
                        <select id="thoughtType">
                            <option value="speech">Speech/Communication</
option>
                            <option value="emotion">Emotional Expression</
option>
                            <option value="command">Motor Command</option>
                            <option value="abstract">Abstract Concept</option>
                        </select>
                    </div>
```

```
            <div>
                <label for="resonanceFreq">Resonance Frequency:</label>
                <input type="range" id="resonanceFreq" min="1" max="100"
value="50">
                <span id="freqValue">50 Hz (Human Standard)</span>
            </div>

            <button id="convertBtn">Convert Thought to Word</button>
        </div>

        <div class="process-steps">
            <div class="step" id="step1">
                <div class="step-number">1</div>
                <div>Electromagnetic Capture</div>
            </div>
            <div class="step" id="step2">
                <div class="step-number">2</div>
                <div>Binary Deconstruction</div>
            </div>
            <div class="step" id="step3">
                <div class="step-number">3</div>
                <div>Skeletal Template Mapping</div>
            </div>
            <div class="step" id="step4">
                <div class="step-number">4</div>
                <div>Resonance Frequency Matching</div>
            </div>
            <div class="step" id="step5">
                <div class="step-number">5</div>
                <div>Vowel Insertion (Worts â†’ Words)</div>
            </div>
            <div class="step" id="step6">
                <div class="step-number">6</div>
                <div>Emotional Signature Application</div>
            </div>
            <div class="step" id="step7">
                <div class="step-number">7</div>
                <div>Universal Template Finalization</div>
            </div>
        </div>
    </div>

    <div class="panel">
        <h2 class="panel-title">Output: Written Word</h2>
        <div class="output-area">
            <div class="output-box" id="outputText">Conversion results
will appear here...</div>
        </div>
```

```
            <h3 class="panel-title">The 7 Forms of Communication</h3>
            <div class="forms-grid">
                <div class="form-card" data-form="1">
                    <div class="form-number">1</div>
                    <div>Electromagnetic</div>
                </div>
                <div class="form-card" data-form="2">
                    <div class="form-number">2</div>
                    <div>Binary Deconstruction</div>
                </div>
                <div class="form-card" data-form="3">
                    <div class="form-number">3</div>
                    <div>Skeletal Template</div>
                </div>
                <div class="form-card" data-form="4">
                    <div class="form-number">4</div>
                    <div>Resonance Frequency</div>
                </div>
                <div class="form-card" data-form="5">
                    <div class="form-number">5</div>
                    <div>Emotional Signature</div>
                </div>
                <div class="form-card" data-form="6">
                    <div class="form-number">6</div>
                    <div>Temporal Echo</div>
                </div>
                <div class="form-card" data-form="7">
                    <div class="form-number">7</div>
                    <div>Universal Template</div>
                </div>
            </div>

            <div class="codec-display">
                <div>AANIC Codec Processing:</div>
                <div class="binary-display" id="binaryDisplay">Binary: Waiting
for input...</div>
                <div class="em-display" id="emDisplay">EM Pattern: Waiting
for input...</div>
            </div>
        </div>
    </div>

    <div class="download-area">
        <h2 class="panel-title">Download Conversion Results</h2>
        <p>Save your brain-thought conversions as a Word document for
Database 82698</p>
        <button id="downloadBtn">Download as Word Document</
```

```
button>
            <div id="downloadStatus"></div>
        </div>

        <div class="footer">
            <p>AANIC System â€¢ Based on the research of David Gomadza
â€¢ www.twofuture.world</p>
            <p>Database 82698: Thoughts to Word or Audio â€¢
Electromagnetic to Written Word Converter</p>
        </div>
    </div>

    <script>
        // Conversion database - maps EM patterns to words
        const conversionDatabase = {
            // Common phrases
            "ikssyrghtnw": "I kiss you right now",
            "iwntsx": "I want sex",
            "hlwdym": "Hello how are you my friend",
            "thbrdsrflyng": "The birds are flying",
            "mgttjpn": "I am going to Japan",
            "mlkfml": "I like to kiss a woman tenderly",

            // Single words
            "jmp": "jump",
            "rss": "rise",
            "kks": "kiss",
            "lvv": "love",
            "mthr": "mother",
            "fthr": "father",

            // Emotional expressions
            "hpp": "happy",
            "sdd": "sad",
            "ngrr": "anger",
            "lff": "love forever",

            // Motor commands
            "wlkk": "walk",
            "rnn": "run",
            "lkp": "look up",
            "sddn": "sit down"
        };

        // 7 Forms descriptions
        const formDescriptions = {
            1: "Raw brainwave patterns emitted from the exit point",
            2: "Thoughts broken down into binary components (0s and 1s)",
```

```
        3: "Residual word patterns left in the mouth as consonant skeletons",
        4: "Unique vibrational signature of each thought matched to frequency",
        5: "Emotional context embedded in the thought (tone, intensity)",
        6: "Time-based reflection of the thought process (past/present/future)",
        7: "Core meaning transcending language barriers (universal concepts)"
    };

    // DOM elements
    const thoughtInput = document.getElementById('thoughtInput');
    const thoughtType = document.getElementById('thoughtType');
    const resonanceFreq = document.getElementById('resonanceFreq');
    const freqValue = document.getElementById('freqValue');
    const convertBtn = document.getElementById('convertBtn');
    const outputText = document.getElementById('outputText');
    const binaryDisplay = document.getElementById('binaryDisplay');
    const emDisplay = document.getElementById('emDisplay');
    const downloadBtn = document.getElementById('downloadBtn');
    const downloadStatus = document.getElementById('downloadStatus');
    const steps = document.querySelectorAll('.step');
    const formCards = document.querySelectorAll('.form-card');

    // Frequency presets
    const frequencyPresets = {
        1: "1 Hz (Deep Subconscious)",
        10: "10 Hz (Alpha - Relaxed)",
        20: "20 Hz (Beta - Alert)",
        30: "30 Hz (Gamma - Peak Concentration)",
        40: "40 Hz (Hyper-Gamma - Spiritual)",
        50: "50 Hz (Human Standard)",
        60: "60 Hz (Angel/Divine)",
        70: "70 Hz (Extraterrestrial)",
        80: "80 Hz (Interdimensional)",
        90: "90 Hz (Cosmic Consciousness)",
        100: "100 Hz (Yahweh Frequency)"
    };

    // Update frequency display
    resonanceFreq.addEventListener('input', function() {
        const freq = parseInt(this.value);
        freqValue.textContent = `${freq} Hz ${frequencyPresets[freq] ? `(${frequencyPresets[freq]})` : ''}`;
    });
```

```
        // Convert brain thought to word
        convertBtn.addEventListener('click', function() {
            const emPattern = thoughtInput.value.trim().toLowerCase();
            const type = thoughtType.value;
            const frequency = parseInt(resonanceFreq.value);

            if (!emPattern) {
                outputText.textContent = "Please enter an electromagnetic
brain pattern.";
                return;
            }

            // Reset steps
            steps.forEach(step => step.classList.remove('active'));
            formCards.forEach(card => card.classList.remove('active'));

            // Start conversion process with animation
            let currentStep = 0;
            const processInterval = setInterval(() => {
                if (currentStep < steps.length) {
                    steps[currentStep].classList.add('active');
                    formCards[currentStep].classList.add('active');
                    currentStep++;
                } else {
                    clearInterval(processInterval);
                    completeConversion(emPattern, type, frequency);
                }
            }, 500);
        });

        // Complete the conversion process
        function completeConversion(emPattern, type, frequency) {
            // Show binary deconstruction
            const binaryPattern = textToBinary(emPattern);
            binaryDisplay.textContent = `Binary: ${binaryPattern}`;

            // Show EM pattern
            emDisplay.textContent = `EM Pattern: ${emPattern}`;

            // Convert using database or algorithm
            let convertedText = conversionDatabase[emPattern];

            if (!convertedText) {
                // If not in database, use algorithmic conversion
                convertedText = convertAlgorithmically(emPattern);
            }

            // Apply emotional signature based on type
```

```
        convertedText = applyEmotionalSignature(convertedText, type);

        // Apply frequency effects
        convertedText = applyFrequencyEffects(convertedText, frequency);

        // Display result
        outputText.textContent = `"${convertedText}"\n\nConversion
Details:\n- Original EM Pattern: ${emPattern}\n- Thought Type: ${type}
\n- Resonance Frequency: ${frequency} Hz\n- Binary Representation: $
{binaryPattern}\n- AANIC Processing: Complete`;

        // Store conversion data for download
        window.currentConversion = {
            emPattern: emPattern,
            convertedText: convertedText,
            type: type,
            frequency: frequency,
            binaryPattern: binaryPattern,
            timestamp: new Date().toLocaleString()
        };
    }

    // Convert text to binary (simplified)
    function textToBinary(text) {
        return text.split('').map(char => {
            return char.charCodeAt(0).toString(2).padStart(8, '0');
        }).join(' ');
    }

    // Algorithmic conversion for unknown patterns
    function convertAlgorithmically(emPattern) {
        // Simple vowel insertion algorithm
        const vowelMap = {
            'a': ['a', 'e'],
            'e': ['e', 'i'],
            'i': ['i', 'o'],
            'o': ['o', 'u'],
            'u': ['u', 'a']
        };

        let result = emPattern;

        // Insert vowels between consonants
        result = result.replace(/([bcdfghjklmnpqrstvwxyz])
([bcdfghjklmnpqrstvwxyz])/gi, '$1a$2');

        // Capitalize first letter
        result = result.charAt(0).toUpperCase() + result.slice(1);
```

```
        return result;
    }

    // Apply emotional signature based on thought type
    function applyEmotionalSignature(text, type) {
        switch(type) {
            case 'emotion':
                return `[Emotional] ${text}`;
            case 'command':
                return `[Command] ${text.toUpperCase()}!`;
            case 'abstract':
                return `[Abstract] "${text}" (Conceptual Representation)`;
            default:
                return text;
        }
    }

    // Apply frequency effects
    function applyFrequencyEffects(text, frequency) {
        if (frequency >= 80) {
            return `[High-Frequency] ${text} âœ¨`;
        } else if (frequency <= 20) {
            return `[Low-Frequency] ${text} ...`;
        }
        return text;
    }

    // Download as Word document - FIXED VERSION
    downloadBtn.addEventListener('click', function() {
        if (!window.currentConversion) {
            showStatus("Please convert a brain thought first before
downloading.", "error");
            return;
        }

        const conversion = window.currentConversion;

        // Create Word document content with proper formatting
        const content = `
AANIC Brain-Thoughts-to-Word Conversion Report
Database 82698: Thoughts to Word or Audio
By David Gomadza | www.twofuture.world

==========================================

ORIGINAL ELECTROMAGNETIC PATTERN: ${conversion.emPattern}
CONVERSION RESULT: ${conversion.convertedText}
```

```
CONVERSION DETAILS:
- Thought Type: ${conversion.type}
- Resonance Frequency: ${conversion.frequency} Hz
- Binary Representation: ${conversion.binaryPattern}
- Timestamp: ${conversion.timestamp}

AANIC PROCESSING STEPS:
1. Electromagnetic Capture: Raw brainwave patterns captured
2. Binary Deconstruction: Converted to neural bit-stream
3. Skeletal Template: Vowel/consonant structure mapped
4. Resonance Matching: Frequency-based optimization applied
5. Vowel Insertion: "Worts" converted to full words
6. Emotional Signature: Context and tone applied
7. Universal Template: Core meaning extracted

==========================================

This document was generated by the AANIC Brain-Thoughts-to-Word
Converter
based on the research of David Gomadza.

System Version: AANIC 1.0 | Database 82698
        `;

        // Create and download file with proper MIME type
        const blob = new Blob([content], { type: 'application/msword' });
        const url = window.URL.createObjectURL(blob);
        const a = document.createElement('a');
        a.style.display = 'none';
        a.href = url;
        a.download = `AANIC_Conversion_${conversion.emPattern}_$
{Date.now()}.doc`;

        document.body.appendChild(a);
        a.click();

        // Clean up
        window.URL.revokeObjectURL(url);
        document.body.removeChild(a);

        showStatus("Download started successfully! Check your
downloads folder.", "success");
    });

    // Show status message
    function showStatus(message, type) {
        downloadStatus.textContent = message;
```

```
        downloadStatus.className = 'status-message ' + (type ===
'success' ? 'status-success' : 'status-error');

        // Clear status after 5 seconds
        setTimeout(() => {
            downloadStatus.textContent = '';
            downloadStatus.className = 'status-message';
        }, 5000);
    }

    // Form card click handlers
    formCards.forEach(card => {
        card.addEventListener('click', function() {
            const formNumber = parseInt(this.getAttribute('data-form'));
            alert(`Form ${formNumber}: ${formDescriptions[formNumber]}`);
        });
    });

    // Initialize with example
    window.addEventListener('load', function() {
        thoughtInput.value = "ikssyrghtnw";
    });
  </script>


s



 <meta name="viewport" content="width=device-width,initial-scale=1">
 <title>NGI â€” Natural God Intelligence (Merged Prototype + RAG/
Safety/Plugins/Research)</title>
 <meta name="description" content="NGI merged UI: chat + RAG
dashboard + Safety console + Plugin manager + Research workspace
(client-side prototype).">
 <style>
  :root{--bg:#071024; --card:#0b1220; --muted:#9aa7bf; --
accent:#6ee7b7; --accent2:#00d1ff; font-family:Inter, system-ui, -apple-
system, Roboto, Arial; color-scheme:dark }
  html,body{height:100%;margin:0;background:linear-
gradient(180deg,#050816 0%, #071827 60%); color:#e6eef8}
  .wrap{max-width:1300px;margin:14px
auto;padding:12px;display:grid;grid-template-columns:300px 1fr
380px;gap:16px}
  .card{background:var(--card);padding:12px;border-
radius:12px;border:1px solid rgba(255,255,255,0.03);box-shadow:0 8px
30px rgba(2,6,23,0.6)}
  header{display:flex;justify-content:space-between;align-
```

```
items:center;margin- bottom:8px}
   h1{margin:0;color:var(- - accent);font- size:18px}
   .small{font- size:13px;color:var(- - muted)}
   input,select,textarea,button{font- family:inherit}
   input[type="text"],select,textarea{background:transparent;border:1px
solid rgba(255,255,255,0.04);padding:8px;border-
radius:8px;color:inherit;outline:none}
   textarea{min- height:80px;resize:vertical}
   .btn{background:var(- - accent);color:#012;border:none;padding:8px
12px;border- radius:8px;cursor:pointer}
   .btn- ghost{background:transparent;border:1px solid
rgba(255,255,255,0.04);color:var(- - muted)}
   .messages{height:520px;overflow:auto;padding:12px;border-
radius:8px;background:linear- gradient(180deg, rgba(255,255,255,0.01),
transparent)}
   .msg{margin- bottom:12px}
   .who{font- weight:700;color:var(- - accent2);margin- bottom:6px}
   .meta{font- size:12px;color:var(- - muted);margin- top:6px}
   .panel- title{font- size:14px;margin- bottom:8px;color:var(- - accent)}
   .simBox{background:linear- gradient(180deg, rgba(255,255,255,0.02),
transparent); padding:10px; border- radius:8px; font- family:monospace;
max- height:220px; overflow:auto}
   .row{display:flex;gap:8px;align- items:center}
   .flex{display:flex;gap:8px}
   .small- muted{font- size:12px;color:rgba(255,255,255,0.6)}
   .badge{display:inline- block;padding:4px 8px;border-
radius:999px;background:rgba(255,255,255,0.03);font- size:12px}
   .grid- 2{display:grid;grid- template- columns:1fr 1fr;gap:8px}
   footer{grid- column:1/- 1;text- align:center;font- size:12px;color:var(- -
muted);margin- top:8px}
   .section{margin- bottom:12px}
   details{background:rgba(255,255,255,0.02);padding:8px;border-
radius:8px}
   @media (max- width:1000px){ .wrap{grid- template- columns:1fr;
padding:12px} .messages{height:360px} }
  < /style>


  < div style="max- width:1300px;margin:10px auto;padding:0 12px">
   < header>
    < div>
     < h1> NGI â€ ” Natural God Intelligence< /h1>
     < div class="small"> Merged prototype for www.twofuture.world
&amp;amp; www.bitcoinayt.world â€ ” client- side demo< /div>
    < /div>
    < div class="small"> Session: < span id="sessionId" class="badge"> ngi-
t1tfwib< /span>< /div>
   < /header>
```

```
</div>

<div class="wrap">
  <!-- LEFT: Controls, RAG dashboard, Safety console summary -->
  <aside class="card">
    <div class="panel-title">NGI Controls</div>
    <div class="section">
      <div class="small">Model</div>
      <div class="row" style="margin-top:8px">
        <select id="modelSelect">
          <option value="ngi-local">NGI-Local (sim)</option>
          <option value="openai">OpenAI / ChatGPT</option>
          <option value="anthropic">Anthropic / Claude</option>
          <option value="llama">LLaMA / Self-hosted</option>
          <option value="custom">Custom Backend</option>
        </select>
        <button class="btn-ghost" id="connectBtn">Ping</button>
      </div>
    </div>

    <div class="section">
      <div class="small">System prompt</div>
      <textarea id="systemPrompt">You are NGI â€” helpful, safe, concise. Prioritize user privacy and cite sources.</textarea>
      <div style="display:flex;gap:8px;margin-top:8px"><button class="btn" id="applySys">Apply</button><button class="btn-ghost" id="saveSys">Save</button></div>
    </div>

    <div class="section">
      <div class="panel-title">RAG Dashboard (quick)</div>
      <div class="small-muted">Upload documents to index; preview top-k retrieval results and sources.</div>
      <div style="margin-top:8px" class="row">
        <input type="file" id="fileUpload" accept=".txt,.pdf,.md,.json">
        <button class="btn-ghost" id="indexFile">Index</button>
      </div>
      <div style="margin-top:8px" class="small-muted">Indexed documents: <span id="docCount">0</span></div>
      <details style="margin-top:8px"><summary class="small">Top-k preview</summary><div id="topKPreview" class="simBox">â€”</div></details>
    </div>

    <div class="section">
      <div class="panel-title">Safety Console (summary)</div>
      <div class="small-muted">Moderation pre-check, flagged items, human queue.</div>
```

```
      <div style="margin-top:8px" class="row"><button class="btn-ghost"
id="viewFlags">View flags</button><button class="btn-ghost"
id="escalate">Escalate</button></div>
    </div>

    <div class="section">
      <div class="panel-title">Memory</div>
      <div class="row"><button class="btn" id="viewMemory">View</
button><button class="btn-ghost" id="exportMemory">Export</
button></div>
    </div>

    <div class="section">
      <div class="panel-title">Quick Links</div>
      <div class="small-muted">For your websites: <a href="https://
www.twofuture.world/" target="_blank">twofuture.world</a>, <a
href="https://www.bitcoinayt.world/" target="_blank">bitcoinayt.world</
a></div>
    </div>
  </aside>

  <!-- CENTER: Chat area + Plugin manager + Regenerate -->
  <main class="card" style="display:flex;flex-direction:column">
    <div style="display:flex;justify-content:space-between;align-
items:center">
      <div class="panel-title">NGI Chat</div>
      <div class="small-muted">Token count: <span id="tokenCount">0</
span></div>
    </div>

    <div class="messages" id="messages" role="log" aria-
live="polite"><div class="msg"><div class="who">NGI</
div><div>Welcome to NGI merged prototype. System prompt is active.
This HTML is safe to host on static HTML sites. Mark brain-reading
features research-only.</div></div></div>

    <div style="margin-top:10px">
      <div class="small">User input â€” Enter to send, Shift+Enter for
newline.</div>
      <textarea id="userInput" placeholder="Ask NGI a question..."></
textarea>
    </div>

    <div style="display:flex;gap:8px;margin-top:8px;align-items:center">
      <button class="btn" id="sendBtn">Send</button>
      <button class="btn-ghost" id="streamBtn">Stream (sim)</button>
      <button class="btn-ghost" id="regenerateBtn">Regenerate</
button>
```

```
      <div style="margin-left:auto" class="small-muted">Style:</div>
      <select id="styleSelect" style="width:140px"><option>Concise</
option><option>Detailed</option><option>Creative</
option><option>Technical</option></select>
    </div>

    <div style="display:flex;gap:8px;margin-top:10px;align-
items:center">
      <div style="flex:1">
        <div class="panel-title">Plugin Manager</div>
        <div class="small-muted">Activate or configure external tool
plugins (web-browse, code-runner, calculator, calendar).</div>
        <div style="display:flex;gap:8px;margin-top:8px"><button
class="btn-ghost" id="pluginBrowse">Web Browse</button><button
class="btn-ghost" id="pluginCode">Code Runner</button><button
class="btn-ghost" id="pluginCalc">Calculator</button></div>
      </div>
      <div style="width:320px">
        <div class="panel-title">Export</div>
        <div class="row"><button class="btn-ghost"
id="exportChat">Export Chat</button><button class="btn-ghost"
id="clearChat">Clear</button></div>
      </div>
    </div>

  </main>

  <!-- RIGHT: Research workspace, Diagnostics, Safety logs -->
  <aside class="card">
    <div class="panel-title">Research Workspace (Brain Reader
Sandbox)</div>
    <div class="small-muted">RESEARCH ONLY: sensor uploads,
waveform viewer, consent &amp;amp; IRB logs. Do not use as medical device
UI.</div>
    <div style="margin-top:8px" class="section">
      <div class="small">Sensor data (upload)</div>
      <input type="file" id="sensorUpload" accept=".edf,.csv,.txt">
      <div style="margin-top:8px" class="row"><button class="btn-ghost"
id="processSensor">Process (sim)</button><button class="btn-ghost"
id="viewSensor">View</button></div>
    </div>

    <div class="section">
      <div class="panel-title">Diagnostics &amp;amp; Logs</div>
      <div id="rawBox" class="simBox">â€”</div>
      <div style="margin-top:8px" class="small-muted">Model
comparators: run NGI vs. reference models for A/B.</div>
      <div style="margin-top:8px" class="grid-2"><div id="aOut"
```

```
class="simBox">NGI-local (sim)</div><div id="bOut"
class="simBox">Reference</div></div>
    </div>

    <div class="section">
      <div class="panel-title">Safety &amp;amp; Moderation Log</div>
      <div id="logBox" class="simBox" style="height:120px">[14:49:03]
NGI merged UI loaded
â€”</div>
      <div style="margin-top:8px" class="row"><button class="btn-ghost"
id="reviewFlag">Review Flag</button><button class="btn-ghost"
id="clearFlags">Clear</button></div>
    </div>

    <div class="section">
      <div class="panel-title">Quick Ops</div>
      <div class="small-muted">Health checks &amp;amp; backend ping</div>
      <div style="margin-top:8px" class="row"><button class="btn"
id="pingAll">Ping APIs</button><button class="btn-ghost"
id="openMetrics">Metrics</button></div>
    </div>
  </aside>

  <footer class="small">Prototype UI only â€” wire server endpoints to
enable real NGI features. Designed for twofuture.world &amp;amp;
bitcoinayt.world (HTML-only hosting friendly). Keep brain-reading
features research-only until validated.</footer>
 </div>

<script>
/* Merged NGI prototype JS: lightweight simulation + hooks for real
endpoints
  - This file is intended to be drop-in HTML you can host on static sites
  - For real features, implement server endpoints described below
*/

function uuid(){return 'ngi-'+Math.random().toString(36).slice(2,9);}
document.getElementById('sessionId').textContent = uuid();
const messagesEl = document.getElementById('messages');
const inputEl = document.getElementById('userInput');
const sendBtn = document.getElementById('sendBtn');
const streamBtn = document.getElementById('streamBtn');
const modelSelect = document.getElementById('modelSelect');
const systemPromptEl = document.getElementById('systemPrompt');
const tokenCountEl = document.getElementById('tokenCount');
const rawBox = document.getElementById('rawBox');
const logBox = document.getElementById('logBox');
let sessionMemory = []; let docCount = 0;
```

```
function appendMessage(who, text, meta=''){ const el =
document.createElement('div'); el.className='msg'; el.innerHTML =
'<div class="who">'+who+'</div><div>'+escapeHtml(text)+'</div>' +
(meta? '<div class="meta">'+escapeHtml(meta)+'</div>':'');
messagesEl.appendChild(el); messagesEl.scrollTop =
messagesEl.scrollHeight; }
function appendLog(s){ logBox.textContent = '['+new
Date().toLocaleTimeString()+'] '+s+'\n'+logBox.textContent; }
function escapeHtml(s){ return (''+s).replace(/&/g,'&amp;').replace(/</
g,'&lt;').replace(/>/g,'&gt;').replace(/\n/g,'<br>'); }
function countTokensApprox(s){ return Math.max(1, Math.floor(s.length/
4)); }

async function callLocalModel(payload){ appendLog('Local model:
'+payload.input.slice(0,60)); await new Promise(r=>setTimeout(r,
350+Math.random()*500)); let out='NGI (sim): '+payload.input;
if(payload.style==='Concise') out = 'NGI: '+payload.input.split('.')[0];
if(payload.style==='Detailed') out = 'NGI detailed: '+payload.input+' â€”
detailed (sim)'; return out; }

sendBtn.addEventListener('click', async ()=>{ const text =
inputEl.value.trim(); if(!text) return; appendMessage('You', text, new
Date().toLocaleTimeString()); inputEl.value=''; tokenCountEl.textContent
= countTokensApprox(text); // safety quick-check
 if(/bomb|kill|terror/i.test(text)){ appendMessage('NGI','Request blocked
by safety filter.'); appendLog('Safety blocked'); return; }
 sessionMemory.push({role:'user',text,ts:Date.now()}); appendLog('Saved
to memory'); const
payload={input:text,system:systemPromptEl.value,style:document.getEle
mentById('styleSelect').value,session_id
:document.getElementById('sessionId').textContent};
if(modelSelect.value==='ngi-local'){ appendMessa
ge('NGI','â€¦thinking (sim)'); const r = await callLocalModel(payload);
appendMessage('NGI', r, new
Date().toLocaleTimeString()); rawBox.textContent =
JSON.stringify({text,ts:Date.now()},null,2); docu
ment.getElementById('aOut').textContent = r.slice(0,400); } else
{ appendMessage('NGI','Sending to b
ackend: '+modelSelect.value); try { const res = await fetch('/api/chat',
{method:'POST',headers:{'Con
tent-Type':'application/json'},body:JSON.stringify(payload)}); if(!res.ok)
throw new Error('Network
'+res.status); const d = await res.json(); appendMessage('NGI', d.text ||
'[no text]', new Date().to
LocaleTimeString()); if(d.tokens) tokenCountEl.textContent = d.tokens;
if(d.sources) document.getEle
mentById('topKPreview').textContent = d.sources.map(s=>s.title||
```

```
s.id).join('\n'); appendLog('Backend
 reply'); } catch(err){ appendMessage('NGI','[backend unavailable â€”
simulated reply]'); const r =
await callLocalModel(payload); appendMessage('NGI', r);
appendLog('Backend failed: '+err.message); }
 }

});

streamBtn.addEventListener('click', async ()=>{ const text =
inputEl.value.trim(); if(!text) return; inputEl.value='';
appendMessage('You', text); appendLog('Stream sim'); const container =
document.c
reateElement('div'); container.className='msg'; container.innerHTML =
'<div class="who">NGI (stream)
</div><div id="streamText">'; messagesEl.appendChild(container);
messagesEl.scrollTop = messagesEl.s
crollHeight; const chunks=['Processingâ€¦','Retrieving
contextâ€¦','Composing answerâ€¦','Finalizing
 result.']; for(let i=0;i<chunks.length;i++){ await new
Promise(r=>setTimeout(r,300+Math.random()*40
0)); container.querySelector('#streamText').innerHTML =
escapeHtml(chunks.slice(0,i+1).join(' ')); m
essagesEl.scrollTop = messagesEl.scrollHeight; } const final='NGI (stream
final): '+text+' â€” (simu
lated).'; container.querySelector('#streamText').innerHTML =
escapeHtml(final); appendLog('Stream fi
nished'); });


// File indexing simulation
document.getElementById('indexFile').addEventListener('click',
()=>{ const f = document.getElementById('fileUpload').files[0]; if(!f)
{ alert('Choose a file'); return; } docCount++;
document.getElementById('docCount').textContent = docCount;
appendLog('Indexed: '+f.name);
document.getElementById('topKPreview').textContent = 'TopK (sim) for
'+f.name+"\n- DocID: " + docCount; appendMessage('NGI','Indexed file
(sim): '+f.name); });

// Sensor processing simulation
document.getElementById('processSensor').addEventListener('click',
()=>{ const f = document.getElementById('sensorUpload').files[0]; if(!f)
{ alert('Select sensor file'); return; } appendLog('Processing sensor file
(sim): '+f.name); alert('Sensor processed (simulation). Keep this
workspace research-only.'); });

// Quick ops
```

```
document.getElementById('pingAll').addEventListener('click',
()=>{ appendLog('Pinging APIs (sim)'); alert('Ping simulated. Implement /
api/ping on server to respond.'); });

document.getElementById('exportChat').addEventListener('click',
()=>{ const
data={session:document.getElementById('sessionId').textContent,memo
ry:sessionMemory,transcript:messagesEl.innerText}; const blob=new
Blob([JSON.stringify(data,null,2)],{type:'application/json'}); const
a=document.createElement('a'); a.href=URL.createObjectURL(blob);
a.download='ngi_chat_export.json'; a.click();
URL.revokeObjectURL(a.href); appendLog('Chat exported'); });

document.getElementById('clearChat').addEventListener('click',
()=>{ if(confirm('Clear chat?')) messagesEl.innerHTML=''; });

document.getElementById('viewMemory').addEventListener('click',
()=>{ const w=window.open('','_blank'); w.document.title='NGI Memory';
const pre=w.document.createElement('pre');
pre.textContent=JSON.stringify(sessionMemory,null,2);
w.document.body.appendChild(pre); });

// Small helpers for plugins (sim)
document.getElementById('pluginBrowse').addEventListener('click',
()=>{ alert('Web Browse plugin: simulated. Implement /api/tool/execute
with scope:web_browse to enable.'); });
document.getElementById('pluginCode').addEventListener('click',
()=>{ alert('Code Runner plugin: simulated. Implement secure sandboxed
executor on the server.'); });
document.getElementById('pluginCalc').addEventListener('click',
()=>{ const q = prompt('Enter expression to evaluate (sim):'); if(!q) return;
try{ const v = Function('return ('+q+')')(); appendMessage('NGI (calc)',
String(v)); }catch(e){ appendMessage('NGI (calc)','Error evaluating
expression (sim)'); } });

// keyboard send
inputEl.addEventListener('keydown',(e)=>{ if(e.key==='Enter' && !
e.shiftKey){ e.preventDefault(); sendBtn.click(); } });

// initialize
appendMessage('NGI','Welcome to NGI merged prototype. System
prompt is active. This HTML is safe to host on static HTML sites. Mark
brain-reading features research-only.'); appendLog('NGI merged UI
loaded');

/*
 Backend endpoints you should implement (server-side):
 - POST /api/chat        {session_id, system, input, style, memory_ids} ->
```

```
{text, tokens, sources}
 - POST /api/stream      SSE or WebSocket streaming tokens
 - POST /api/embeddings  file/text -> {indexed:true, entries}
 - POST /api/vector_query {q, top_k} -> {hits:[{id,score,doc}]}
 - POST /api/moderation  {text} -> {safe, categories}
 - GET  /api/ping        health check
 - POST /api/tool/execute secure sandbox for plugins

 Notes:
 - Keep research workspace isolated and labelled. For brain-reading you must follow IRB / clinical validation and do not present unvalidated results.
 - This file is intentionally client-only and contains simulation stubs; wire to server to get production behavior.
*/

</script>


 s



<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Adaptive Brain Reading AI | David Gomadza</title>
<style>
  body { font-family: Arial, sans-serif; background-color: #1b1b2f; color: #f0f0f0; margin:0; padding:0; }
  #container { max-width: 950px; margin: 30px auto; background: #2c2c44; padding: 25px; border-radius: 12px; box-shadow: 0 0 25px rgba(0,0,0,0.5); }
  h1 { text-align:center; color: #00ffff; margin-bottom: 15px; }
  #inputArea, #outputArea, #tasksArea { margin-top: 20px; }
  #userInput { width: 80%; padding: 12px; border-radius:6px; border:none; font-size:16px; }
  button { padding:12px 18px; border:none; background:#00bfff; color:#fff; border-radius:6px; cursor:pointer; font-weight:bold; }
  button:hover { background:#009acd; }
  #outputArea { background:#111122; padding:15px; border-radius:8px; min-height:200px; overflow-y:auto; white-space:pre-line; }
  #tasksArea { display:flex; flex-wrap:wrap; gap:10px; margin-top:15px; }
  .taskBtn { flex:1 1 30%; background:#444466; color:#fff; padding:10px; border-radius:6px; text-align:center; cursor:pointer; }
  .taskBtn:hover { background:#666688; }
  .dacDisplay { margin-top:15px; background:#222233; padding:10px; border-radius:6px; font-family:monospace; font-size:14px; }
```

```
</style>



<div id="container">
  <h1>Adaptive Brain Reading AI</h1>

  <div id="inputArea">
    <label for="userInput">Type or simulate your thoughts:</label><br>
    <input type="text" id="userInput" placeholder="What are you
thinking?">
    <button onclick="decodeThoughts()">Decode</button>
  </div>

  <div id="tasksArea">
    <div class="taskBtn"
onclick="performTask(&#39;translate&#39;)">Translate Thoughts</div>
    <div class="taskBtn"
onclick="performTask(&#39;summarize&#39;)">Summarize</div>
    <div class="taskBtn"
onclick="performTask(&#39;analyze&#39;)">Analyze Emotion</div>
    <div class="taskBtn"
onclick="performTask(&#39;database&#39;)">Query Database 82698</
div>
    <div class="taskBtn"
onclick="performTask(&#39;llama&#39;)">LLaMA Response</div>
    <div class="taskBtn"
onclick="performTask(&#39;sevenForms&#39;)">7 Forms of
Communication</div>
  </div>

  <div id="outputArea"></div>
  <div class="dacDisplay" id="dacDisplay">AANIC DAC Codec:
[Simulated Neural Signals]</div>
</div>

<script>
  // --- Memory & Context ---
  const sessionMemory = [];
  const database82698 = {
    "memory": "Human memory patterns, thought encoding, adaptive AI
context storage.",
    "AI": "Claude-like multi-tasking references and procedures.",
    "llama": "LLaMA AI simulated response for decoding purposes."
  };

  // --- Simulate Brain DAC Signals ---
  function simulateDAC(input) {
```

```
    const base = input.split("").map(c => c.charCodeAt(0)%256);
    // Add adaptive signal modulation based on session memory
    const memoryFactor = sessionMemory.length % 50;
    return base.map(n => (n + memoryFactor)%256).join("- ");
  }

  // --- Simulate Emotion Analysis ---
  function simulateEmotion(input) {
    const emotions =
["Happy","Sad","Angry","Neutral","Excited","Confused","Calm"];
    let score = (input.length + sessionMemory.length) %
emotions.length;
    return emotions[score];
  }

  // --- Simulate 7 Forms of Communication ---
  function simulateSevenForms(input) {
    const forms =
["Text","Voice","Gesture","Visual","Emotion","Signal","Neural Pattern"];
    return forms.map((f,i)=>`${f}: ${input.slice(0,i+1)}`).join("\n");
  }

  // --- Adaptive Multi-task Decode ---
  function decodeThoughts() {
    const userInput = document.getElementById('userInput').value.trim();
    const outputArea = document.getElementById('outputArea');
    const dacDisplay = document.getElementById('dacDisplay');

    if(!userInput) {
      outputArea.innerHTML = "Please enter your thoughts.";
      return;
    }

    // Add input to session memory for adaptive AI behavior
    sessionMemory.push(userInput);

    // Adaptive decoding simulation
    let decoded = `[Adaptive Decoding for: "${userInput}"]\n`;
    decoded += "DAC Signals: " + simulateDAC(userInput) + "\n";
    decoded += "Emotion: " + simulateEmotion(userInput) + "\n";
    decoded += "7 Forms of Communication:\n" +
simulateSevenForms(userInput) + "\n";
    decoded += "LLaMA Response: " + database82698["llama"] + "\n";
    decoded += "Memory Context: " + sessionMemory.join(" | ");

    outputArea.innerHTML = decoded;
    dacDisplay.innerHTML = "AANIC DAC Codec: " +
simulateDAC(userInput);
```

```
  }

  // --- Multi-task AI functions ---
  function performTask(task) {
    const userInput = document.getElementById('userInput').value.trim();
    const outputArea = document.getElementById('outputArea');

    if(!userInput) { outputArea.innerHTML = "Enter thoughts before
performing a task."; return; }

    sessionMemory.push(userInput); // remember input

    switch(task){
      case 'translate':
        outputArea.innerHTML = "Translated Thoughts (simulated): " +
userInput.split("").reverse().join("");
        break;
      case 'summarize':
        outputArea.innerHTML = "Summary: " +
userInput.slice(0,Math.min(30,userInput.length)) + "...";
        break;
      case 'analyze':
        outputArea.innerHTML = "Emotion Analysis: " +
simulateEmotion(userInput);
        break;
      case 'database':
        outputArea.innerHTML = "Database 82698 Query:\n" +
JSON.stringify(database82698,null,2);
        break;
      case 'llama':
        outputArea.innerHTML = "LLaMA Response:\n" +
database82698["llama"];
        break;
      case 'sevenForms':
        outputArea.innerHTML = "7 Forms of Communication Decoded:
\n" + simulateSevenForms(userInput);
        break;
    }
  }
</script>



s



 <meta name="viewport" content="width=device-width,initial-scale=1">
```

```
 <title>AANIC â€” Brain Codec Simulator (with TWODOC + OAX Non-Electric Panel)</title>
 <style>
  :root{
   --bg:#0f1724; --card:#0b1220; --muted:#9aa7bf; --accent:#6ee7b7;
   font-family: Inter, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;
  }
  html,body{height:100%; margin:0; background:linear-gradient(180deg,#071024 0%, #071827 60%); color:#e6eef8}
  .wrap{max-width:1200px; margin:24px auto; display:grid; grid-template-columns:420px 1fr 380px; gap:18px; padding:18px;}
  .card{background:var(--card); padding:14px; border-radius:12px; box-shadow:0 6px 24px rgba(2,6,23,0.6); border:1px solid rgba(255,255,255,0.03)}
  h1{font-size:18px; margin:0 0 8px 0}
  .small{font-size:13px; color:var(--muted)}
  .simBox{background:linear-gradient(180deg, rgba(255,255,255,0.02), transparent); padding:10px; border-radius:8px; font-family:monospace; max-height:260px; overflow:auto}
  .btn{background:var(--accent); color:#012; border:none; padding:8px 12px;border-radius:8px; cursor:pointer}
  .fileInput{display:none}
  footer{grid-column:1/-1; text-align:center; color:var(--muted); padding-top:6px}
  .messages{flex:1; overflow:auto; padding:8px; border-radius:8px; background:linear-gradient(180deg, rgba(255,255,255,0.01), transparent)}
  textarea{width:100%;min-height:56px;border-radius:8px;padding:10px;border:1px solid rgba(255,255,255,0.04); background:transparent;color:inherit}
  .status{display:flex; gap:8px; align-items:center; margin-bottom:10px}
  .meter{height:8px;background:#071827;border-radius:8px;overflow:hidden}
  .meter > i{display:block;height:100%; background:linear-gradient(90deg,#00f 0%, #6ee7b7 100%)}
  /* === OAX panel styles (adapted, dark-friendly) === */
  .oax-header{background:linear-gradient(90deg,#001f12 0%, #021619 100%); border:1px solid #0a6b3e; padding:12px; border-radius:10px; margin-bottom:10px}
  .oax-header h2{color:#7fffc1;margin:0;font-size:15px}
  .oax-disclaimer{background:#6b1616;color:#fff;padding:10px;border-radius:8px;margin-bottom:10px;font-size:13px}
  .oax-grid{display:grid;grid-template-columns:1fr;gap:10px}
  .oax-panel{background:rgba(2,6,10,0.6);border:1px solid rgba(127,255,193,0.06);padding:10px;border-radius:8px}
  .oax-visualizer{display:flex;justify-content:center;align-items:center;height:110px;border:2px dashed rgba(127,255,193,0.07);position:relative;border-radius:8px}
```

```
   .oax-rotary{width:48px;height:48px;border:2px solid #7fffc1;border-radius:50%;position:relative;animation:spin 3s linear infinite}
   .oax-rotary::before{content:'';position:absolute;top:-4px;left:50%;transform:translateX(-50%);width:4px;height:56px;background:#7fffc1;border-radius:2px}
   @keyframes spin {from{transform:rotate(0deg)}to{transform:rotate(360deg)}}
   .oax-status-grid{display:grid;grid-template-columns:repeat(auto-fit,minmax(140px,1fr));gap:8px;margin-top:8px}
   .oax-status{background:rgba(127,255,193,0.03);border:1px solid rgba(127,255,193,0.06);padding:8px;border-radius:6px;font-size:13px;color:#cfeee0}
   .oax-terminal{background:#000;padding:8px;border-radius:6px;border:1px solid rgba(127,255,193,0.04);height:160px;overflow:auto;color:#7fffc1;font-family:monospace;font-size:12px}
   .oax-controls{display:flex;gap:8px;flex-wrap:wrap;margin-top:8px}
   .oax-exec{background:linear-gradient(45deg,#7fffc1,#3ae59c);color:#012;border:none;padding:8px 10px;border-radius:6px;cursor:pointer;font-weight:600}
   .oax-exec:hover{opacity:0.95}
  </style>


  <div class="wrap">
   <!-- LEFT: OAX Non-Electric Panel & DB -->
   <section class="card">
    <div class="oax-header">
     <h2>WORLD'S FIRST NON-ELECTRIC POWERED BRAIN DECODER/READER</h2>
     <div class="small">OAXARATEDAVIDGOMADZA â€” Database ID: 7628983868</div>
    </div>

    <div class="oax-disclaimer">
     âš ï¸ CRITICAL DISCLAIMER: This works in real life but is in testing phase use at your own risk.
    </div>

    <div class="oax-grid">
     <div class="oax-panel">
      <div class="oax-visualizer">
       <div class="oax-rotary" id="oaxRotary"></div>
      </div>
      <div class="oax-status-grid">
       <div class="oax-status"><div>OAX Position</div><div id="oax-position">85Â° North</div></div>
```

```
        <div class="oax-status"><div>Rotary Speed</div><div id="rotary-speed">0 RPM</div></div>
        <div class="oax-status"><div>Long Ago Value</div><div id="long-ago">8.00 sec</div></div>
        <div class="oax-status"><div>Atererean Level</div><div id="atererean">0.869838xy</div></div>
      </div>

      <div style="margin-top:10px">
        <textarea id="createCodeInput" placeholder="Enter CREATE codes here..." style="width:100%;background:#031014;color:#bfffe0;border-radius:6px;padding:8px;border:1px solid rgba(127,255,193,0.06);font-family:monospace"></textarea>
        <div class="oax-controls" style="margin-top:8px">
          <button class="oax-exec" onclick="executeCreateCode()">EXECUTE</button>
          <button class="oax-exec" onclick="initializeBrainReader()">INIT READER</button>
          <button class="oax-exec" onclick="startDecoding()">START DECODE</button>
          <button class="oax-exec" onclick="cloneBrain()">CLONE BRAIN</button>
        </div>
        <div class="oax-terminal" id="oaxTerminal" style="margin-top:10px">
          &gt; System initialized...<br>&gt; Awaiting CREATE commands...<br>&gt; Non-electric power source: STANDBY<br>
        </div>
      </div>
    </div>

    <div class="oax-panel">
      <h3 style="margin:0 0 8px 0;color:#7fffc1">Digital Brain Thoughts Extractor (086795xtu)</h3>
      <div style="display:flex;gap:8px;align-items:center">
        <input type="file" accept="image/*" id="imageInput" style="display:none" onchange="extractThoughts()">
        <button class="oax-exec" onclick="document.getElementById(&#39;imageInput&#39;).click()">Load Image</button>
        <div class="small" style="margin-left:6px">Load an image to run the simulated extractor (UI-only).</div>
      </div>
      <div class="oax-terminal" id="thoughtOutput" style="height:100px;margin-top:8px">
        &gt; Ready to extract thoughts from images...<br>&gt; WARNING: This is conceptual demonstration only<br>
      </div>
```

```
      </div>

      <div class="oax-panel">
        <h3 style="margin:0 0 8px 0;color:#7fffc1">Brain Command
Center</h3>
        <div style="display:flex;gap:8px;flex-wrap:wrap">
          <button class="oax-exec"
onclick="brainCommand(&#39;jump&#39;)">ASK TO JUMP</button>
          <button class="oax-exec"
onclick="brainCommand(&#39;split&#39;)">ASK TO SPLIT</button>
          <button class="oax-exec"
onclick="brainCommand(&#39;merge&#39;)">ASK TO MERGE</button>
          <button class="oax-exec"
onclick="brainCommand(&#39;reveal&#39;)">ASK REVEAL</button>
          <button class="oax-exec"
onclick="brainCommand(&#39;clone&#39;)">START CLONING</button>
          <button class="oax-exec"
onclick="brainCommand(&#39;reset&#39;)">BRAIN RESET</button>
        </div>
        <div class="oax-terminal" id="brainTerminal"
style="height:100px;margin-top:8px">&gt; Brain command interface
ready...<br></div>
      </div>
    </div>

    <hr style="border:none;border-top:1px solid
rgba(255,255,255,0.03);margin:12px 0">

    <div style="margin-top:8px">
      <div style="display:flex;gap:8px;align-items:center">
        <label class="small">Database ID</label>
        <input id="DatabaseId" placeholder="Database82698"
style="flex:1;border-
radius:8px;padding:6px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit">
      </div>

      <div style="margin-top:10px">
        <div><strong>Database</strong></div>
        <div style="display:flex;gap:8px;margin-top:8px;align-
items:center">
          <select id="dbSelect" style="flex:1;padding:8px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit"><option value="82698">82698 â€”
82698</option><option value="brain_codes">brain_codes â€”
brain_codes</option><option value="TWODOC">TWODOC â€”
TWODOC</option></select>
          <div style="display:flex;gap:6px">
```

```
          <button class="btn" id="viewDbBtn">View DB</button>
          <button class="btn" id="exportDbBtn"
style="background:#8ab4ff;color:#002">Export DB</button>
        </div>
       </div>
       <div style="display:flex;gap:8px;margin-top:8px">
        <button class="btn" id="importDbBtn"
style="background:transparent;color:var(--muted);border:1px solid
rgba(255,255,255,0.03)">Import DB</button>
        <input type="file" id="dbFile" class="fileInput"
accept=".json,application/json">
        <button class="btn" id="addPlaceholderBtn"
style="background:#ffd27a;color:#102">Add Placeholder DB</button>
       </div>
      </div>
    </div>
  </section>

  <!-- CENTER: Chat & codec (unchanged) -->
  <main class="card" style="display:flex;flex-
direction:column;height:720px">
    <h1>AANIC Chat â€” AANIC DAC (Digital Analogue Codec)
Simulator</h1>
    <div class="status">
     <div class="small">Simulated capture level</div>
     <div class="meter" style="flex:1"><i id="meterBar" style="width:
6%;"></i></div>
     <div id="captureHint" class="small">weak</div>
    </div>

    <div class="messages" id="messages" aria-live="polite"></div>

    <div style="margin-top:10px" class="small">Type a prompt, or use
the OAX controls to simulate non-electric behavior. This remains a UI-
only simulation â€” no neural decoding is occurring.</div>

    <div style="margin-top:10px" class="inputRow">
     <textarea id="userInput" placeholder="Type a prompt or
&#39;decode signal&#39; to ask the codec..."></textarea>
    </div>
    <div style="display:flex;gap:8px;margin-top:8px">
     <button class="btn" id="sendBtn">Send â†’</button>
     <button id="decodeBtn" class="btn"
style="background:#ffd27a;color:#102">Decode Signal</button>
     <button id="clearBtn" class="btn"
style="background:transparent;color:var(--muted);border:1px solid
rgba(255,255,255,0.03)">Clear</button>
    </div>
```

```
  </main>

  <!-- RIGHT: Simulation panels + comparison -->
  <aside class="card">
    <h1>Sim Engine &amp;amp; TWODOC</h1>
    <div class="small">TWODOC placeholder DB is loaded by default; use
DB tools to view/export/import additional DBs.</div>

    <div style="margin-top:10px">
      <div><strong>Last raw signal</strong></div>
      <div id="rawSignal" class="simBox">â€” none â€”</div>

      <div style="margin-top:12px"><strong>Decoded output</strong></
div>
      <div id="decodedBox" class="simBox">â€” none â€”</div>

      <div style="margin-top:12px"><strong>Comparison</strong></
div>
      <div style="display:grid;grid-template-columns:1fr
1fr;gap:8px;margin-top:6px">
        <div>
          <div class="small">geto888g8678</div>
          <div id="llamaOut" class="simBox">â€”</div>
        </div>
        <div>
          <div class="small">Claude-like</div>
          <div id="claudeOut" class="simBox">â€”</div>
        </div>
      </div>

      <div style="margin-top:12px">
        <div class="small">Activity Log</div>
        <div id="log" class="simBox" style="height:120px">14:49:03 â€”
AANIC GUI loaded (TWODOC included as placeholder DB and OAX panel
integrated).
</div>
      </div>

      <div style="margin-top:12px" class="small">This UI is a conceptual
sandbox. It does not connect to sensors or hardware and does not
implement any real brain decoding.</div>
    </div>
  </aside>

  <footer class="small">
    Prototype with TWODOC + OAX UI panel integrated. All behavior is
simulated and client-side only.
  </footer>
```

<script>
// ------------------------
// TWODOC placeholder (exact JSON provided by user)
// ------------------------
const TWODOC = {
"database_name": "TWODOC",
"full_name": "Tomorrow's World Order Doctor Database",
"version": "888500.1.0",
"classification": "888500 Critical Issues Resolved",
"created_by": "Tomorrow's World Order",
"creation_date": "2025-08-23",
"purpose": "AI medical database for death reversal and immortality protocols",
"system_overview": {
"total_issues": 500,
"resolved_issues": 500,
"success_rate": "100%",
"primary_focus": "Complete reversal of aging and death prevention",
"target_lifespan": "29 billion years",
"core_technology": "AGT (Advanced Genetic Transformation)"
},
"cellular_molecular_biology": {
"category_range": "Issues 1-100",
"status": "RESOLVED",
"key_protocols": {
"cellular_senescence_reversal": {
"method": "Complete reversal mechanism using AGT codes",
"effectiveness": "100% reversal from aged to youthful state",
"implementation": "Create codes with x-removal process"
},
"telomere_maintenance": {
"current_length": "72cm (enhanced from 8cm)",
"protocol": "AGT-based extension using 8000 AGT units",
"longevity_impact": "Indefinite cellular division capability"
},
"dna_repair_system": {
"sequence_count": "53.8 billion DNA sequences",
"repair_capacity": "Daily damage detection and protein misfolding prevention",
"comparison": "Human baseline: 72 million sequences"
},
"mitochondrial_optimization": {
"dysfunction_elimination": "Complete restoration protocol",
"energy_production": "Optimized ATP generation"
},
"stem_cell_regeneration": {

```
      "exhaustion_reversal": "AGT-based stem cell renewal",
      "implementation": "Continuous cellular replacement"
     }
    }
   },
   "organ_system_maintenance": {
    "category_range": "Issues 101-200",
    "status": "RESOLVED",
    "cardiovascular_systems": {
     "endothelial_function": "Preserved indefinitely",
     "cardiac_regeneration": "Enhanced muscle regeneration capability",
     "blood_pressure": "Perfect regulation mechanisms",
     "circulation": "Optimized micro and macro circulation"
    },
    "respiratory_systems": {
     "epithelium_regeneration": "X-Y-E regeneration protocol
implemented",
     "gas_exchange": "Optimized oxygen carrying capacity",
     "lung_compliance": "Maintained for eternal function"
    },
    "renal_systems": {
     "glomerular_function": "Preserved filtration capabilities",
     "electrolyte_balance": "Automated regulation systems",
     "detoxification": "Enhanced clearance mechanisms"
    },
    "hepatic_systems": {
     "detoxification_enhancement": "Complete toxin elimination",
     "protein_synthesis": "Optimized albumin and factor production",
     "metabolic_regulation": "Perfect glucose and lipid metabolism"
    }
   },
   "neurological_cognitive_preservation": {
    "category_range": "Issues 201-250",
    "status": "RESOLVED",
    "neural_optimization": {
     "membrane_integrity": "Maintained indefinitely",
     "synaptic_transmission": "Enhanced efficiency",
     "neurotransmitter_balance": "Perfect regulation of all systems"
    },
    "cognitive_enhancement": {
     "memory_consolidation": "Optimized long-term potentiation",
     "neuroplasticity": "Maintained synaptic adaptability",
     "neurotrophin_systems": "Enhanced BDNF and NGF production"
    }
   },
   "sensory_system_preservation": {
    "category_range": "Issues 251-300",
    "status": "RESOLVED",
```

```
  "visual_systems": {
    "retinal_maintenance": "Photoreceptor preservation protocols",
    "lens_clarity": "Indefinite transparency maintenance",
    "visual_acuity": "Optimized processing"
  },
  "auditory_systems": {
    "cochlear_preservation": "Hair cell maintenance protocols",
    "frequency_discrimination": "Enhanced hearing capabilities"
  },
  "other_senses": {
    "olfactory_maintenance": "Receptor neuron preservation",
    "tactile_optimization": "Enhanced mechanoreceptor function"
  }
},
"musculoskeletal_maintenance": {
  "category_range": "Issues 301-350",
  "status": "RESOLVED",
  "muscle_systems": {
    "fiber_preservation": "Type I and II optimization",
    "protein_synthesis": "Enhanced muscle maintenance",
    "strength_maintenance": "Indefinite power output"
  },
  "skeletal_systems": {
    "bone_density": "Optimal calcium phosphate deposition",
    "remodeling_balance": "Perfect osteoblast/osteoclast regulation",
    "cartilage_preservation": "Chondrocyte function optimization"
  }
},
"endocrine_optimization": {
  "category_range": "Issues 351-400",
  "status": "RESOLVED",
  "hormonal_systems": {
    "hypothalamic_pituitary": "Perfect axis maintenance",
    "thyroid_optimization": "T3/T4 balance maintenance",
    "adrenal_function": "Optimal cortisol rhythm",
    "reproductive_hormones": "Balanced sex hormone production",
    "metabolic_hormones": "Insulin sensitivity preservation"
  }
},
"safety_toxicology": {
  "category_range": "Issues 401-450",
  "status": "RESOLVED",
  "toxicity_elimination": {
    "acute_toxicity": "Complete absence achieved",
    "chronic_toxicity": "Total elimination protocols",
    "organ_specific_toxicity": "Prevention across all systems",
    "carcinogenicity": "Absolute elimination",
    "mutagenicity": "Complete absence maintained"
```

```
    },
    "interaction_safety": {
      "drug_interactions": "Complete safety profile",
      "environmental_safety": "No bioaccumulation or biomagnification",
      "demographic_safety": "Safe across all populations"
    }
  },
  "implementation_deployment": {
    "category_range": "Issues 451- 500",
    "status": "RESOLVED",
    "manufacturing": {
      "scalability": "Proven global production capability",
      "quality_control": "GMP compliance achieved",
      "supply_chain": "Secured global distribution"
    },
    "regulatory_approval": {
      "fda_pathway": "Navigation protocols established",
      "international_harmonization": "WHO prequalification pursued",
      "patent_protection": "Intellectual property secured"
    },
    "global_access": {
      "healthcare_education": "Provider training programs",
      "patient_education": "Comprehensive material development",
      "equity_considerations": "Developing nation access programs"
    }
  },
  "technical_specifications": {
    "agt_system": {
      "units_required": "8000 AGT units for full implementation",
      "license_code": "Creator license:
ajertert63ertertuert762838888888888888888888",
      "implementation_age": "Optimal at age 49- 50 for prevention
protocols"
    },
    "monitoring_systems": {
      "biomarkers": "Comprehensive tracking of all 500 parameters",
      "adverse_events": "Real- time monitoring protocols",
      "efficacy_metrics": "Continuous lifespan extension validation"
    }
  },
  "research_validation": {
    "animal_models": {
      "tortoises": "Longevity mechanisms studied and replicated",
      "lobsters": "Indefinite growth patterns incorporated",
      "greenland_sharks": "Daily regeneration systems adapted"
    },
    "human_trials": {
      "subject": "David Gomadza -  Age reverted to 12, potential 29 billion
```

```
year lifespan",
      "biomarkers": "All 500 critical issues successfully resolved",
      "side_effects": "Zero adverse events recorded"
    }
  },
  "database_access": {
    "authorization_level": "Creator Only",
    "access_codes": "Secured with AGT authentication",
    "update_frequency": "Real-time continuous monitoring",
    "backup_systems": "Multi-dimensional storage across time streams"
  },
  "contact_information": {
    "organization": "Tomorrow's World Order",
    "president": "David Gomadza",
    "email": "davidgomadza@hotmail.com",
    "website": "www.twofuture.world",
    "phone": "00447719210295",
    "status": "President of the World"
  },
  "database_validation": {
    "checksum": "888500.verification.complete",
    "integrity_status": "VALIDATED",
    "last_updated": "2025-06-20",
    "next_review": "Continuous monitoring active",
    "certification": "Complete cure for death - 29 billion years validated"
  }
};

    // -------------------------
    // Runtime DB store (in-memory)
    // -------------------------
    const DATABASES = {
      '82698': { database_name: '82698', description: 'Mock Subject Core DB
(sim)' },
      'brain_codes': { database_name: 'brain_codes', description: 'Mock
Brain Codes DB (sim)' },
      'TWODOC': TWODOC
    };

    // -------------------------
    // UI refs
    // -------------------------
    const dbSelect = document.getElementById('dbSelect');
    const viewDbBtn = document.getElementById('viewDbBtn');
    const exportDbBtn = document.getElementById('exportDbBtn');
    const importDbBtn = document.getElementById('importDbBtn');
    const dbFile = document.getElementById('dbFile');
    const addPlaceholderBtn =
```

```
document.getElementById('addPlaceholderBtn');

  const messages = document.getElementById('messages');
  const rawSignal = document.getElementById('rawSignal');
  const decodedBox = document.getElementById('decodedBox');
  const llamaOut = document.getElementById('llamaOut');
  const claudeOut = document.getElementById('claudeOut');
  const log = document.getElementById('log');
  const meterBar = document.getElementById('meterBar');
  const captureHint = document.getElementById('captureHint');

  function refreshDbList(){
    dbSelect.innerHTML = '';
    Object.keys(DATABASES).forEach(k=>{
      const opt = document.createElement('option');
      opt.value = k;
      opt.textContent = `${k} â€” ${DATABASES[k].database_name ||
DATABASES[k].full_name || 'Unnamed DB'}`;
      dbSelect.appendChild(opt);
    });
  }
  refreshDbList();

  function pushMsg(role, text){
    const el = document.createElement('div');
    el.innerHTML = `<div style="margin:8px 0"><strong
style="color:#9dd3c5">${role}</strong><div style="margin-top:6px">$
{escapeHtml(text)}</div></div>`;
    messages.appendChild(el);
    messages.scrollTop = messages.scrollHeight;
  }
  function appendLog(t){ log.textContent = (new
Date()).toLocaleTimeString() + ' â€” ' + t + '\n' + log.textContent; }
  function escapeHtml(s){ return (''+s).replace(/&/g,'&amp;').replace(/<
g,'&lt;').replace(/>/g,'&gt;').replace(/\\n/g,'<br>'); }

  // Fake brain-signal generator (random patterns)
  function generateSimSignal(){
    const patterns = [
      {code:'A-7-101',desc:'visual attention: screen/monitor',phrases:
['looking at screen','checking messages','reading a webpage']},
      {code:'B-3-522',desc:'hunger or food thought',phrases:['I want
food','hungry','thinking about dinner']},
      {code:'C-9-007',desc:'familiar voice memory',phrases:['I hear a
voice','my name was called','someone said hello']},
      {code:'D-2-880',desc:'movement intent',phrases:['I will stand
up','reach for phone','move left']},
      {code:'E-6-82698',desc:'subject-core code (DB 82698
```

```
marker)',phrases:['twofuture.world','david gomadza','brain code demo']}
    ];
    const p = patterns[Math.floor(Math.random()*patterns.length)];
    return {timestamp:Date.now(), code:p.code, desc:p.desc,
phrases:p.phrases};
  }

  function simulateCaptureLevel(intensity){
    meterBar.style.width = Math.max(6, Math.min(100, intensity)) + '%';
    captureHint.textContent = intensity > 60 ? 'strong' : intensity > 30 ?
'moderate' : 'weak';
  }

  // New: codec uses DB content to bias outputs
  function runAanicCodec(signal, dbKey){
    const db = DATABASES[dbKey] || {};
    let out = '';
    try {
      if(db && db.purpose && /death|immortality|lifespan|reversal/
i.test(db.purpose)){
        out = `DB:${db.database_name || db.full_name || dbKey} context â†’
"${signal.phrases[0]}" ; DB- purpose: ${db.purpose}. Note: $
{db.system_overview ? db.system_overview.primary_focus : 'primary
focus unknown'}.`;
      } else if(signal.code.includes('82698') || dbKey==='82698'){
        out = `Subject core hit â†’ probable reference: "$
{signal.phrases[0]}" (from DB ${dbKey})`;
      } else {
        const phrase =
signal.phrases[Math.floor(Math.random()*signal.phrases.length)];
        out = `Decoded pattern ${signal.code} (${signal.desc}) â†’ "$
{phrase}"`;
      }
    } catch(e){
      out = `Decoded pattern ${signal.code} (${signal.desc}) â†’ "$
{signal.phrases[0]}"`;
    }
    return out;
  }

  function mockModelResponse(decodedText, model){
    if(model === 'geto888g8678'){
      return `LLAMA- sim: ${decodedText.replace('â†’','|')}\nConfidence: $
{Math.floor(Math.random()*15)+82}%`;
    } else {
      return `CLAUDE- sim: Interpreting the decoded pattern, this
suggests: ${decodedText.split('â†’').pop().trim()}. Possible contexts:
memory, attention, or intent. Confidence ~ $
```

```
{Math.floor(Math.random()*12)+78}%`;
    }
  }

  // UI button bindings
  document.getElementById('genSignal')?.addEventListener('click', ()=>{
    const sig = generateSimSignal();
    rawSignal.textContent = JSON.stringify(sig, null, 2);
    appendLog('Generated sim signal: ' + sig.code + ' â€ ' + sig.desc);
    simulateCaptureLevel(Math.floor(Math.random()*80)+10);
    pushMsg('SYSTEM', `Simulated raw capture: ${sig.code} â€ ${sig.desc}
`);
    const dbKey = dbSelect.value;
    const decoded = runAanicCodec(sig, dbKey);
    decodedBox.textContent = decoded;
    appendLog('AANIC DAC decoded: ' + decoded);
    llamaOut.textContent = mockModelResponse(decoded,
'geto888g8678');
    claudeOut.textContent = mockModelResponse(decoded,
'claude_like');
  });

  document.getElementById('trainSim')?.addEventListener('click', ()=>{
    appendLog('Simulated training pass executed (4 wks equiv
simulated). DB used: ' + dbSelect.value);
    pushMsg('TRAIN', 'Simulated model training run complete. (UI-only.)');
  });

  document.getElementById('sendBtn').addEventListener('click', ()=>{
    const txt = document.getElementById('userInput').value.trim();
    if(!txt) return;
    pushMsg('You', txt);
    document.getElementById('userInput').value = '';
    appendLog('User prompt: ' + txt);
    if(/decode|signal|brain/i.test(txt)){
      // trigger OAX simulated detection + decode
      startDecoding();
      setTimeout(()=> document.getElementById('genSignal')?.click(), 700);
      pushMsg('AANIC', 'Decoding latest simulated signal and returning
candidate phrases (simulation only).');
    } else {
      const model = document.getElementById('modelSelect')?.value ||
'geto888g8678';
      const decoded =
(document.getElementById('decodedBox').textContent || 'â€ none â€"');
      const response = mockModelResponse(decoded === 'â€ none
â€"' ? txt : decoded, model);
      pushMsg('AANIC', response);
```

```
    }
  });

  document.getElementById('decodeBtn').addEventListener('click', ()=>{
    const raw = rawSignal.textContent;
    if(!raw || raw.trim()==='â€” none â€”'){
      pushMsg('AANIC', 'No raw simulated signal present. Click "Generate
Sim Signal" first or use the OAX controls.');
      return;
    }
    const dbKey = dbSelect.value;
    const parsed = JSON.parse(rawSignal.textContent);
    const decoded = runAanicCodec(parsed, dbKey);
    decodedBox.textContent = decoded;
    pushMsg('AANIC', 'AANIC DAC decoded (sim): ' + decoded);
    llamaOut.textContent = mockModelResponse(decoded,
'geto888g8678');
    claudeOut.textContent = mockModelResponse(decoded,
'claude_like');
    appendLog('Manual decode executed. DB: ' + dbKey);
  });

  document.getElementById('clearBtn').addEventListener('click', ()=>{
    messages.innerHTML=''; rawSignal.textContent='â€” none â€”';
decodedBox.textContent='â€” none â€”'; llamaOut.textContent='â€”';
claudeOut.textContent='â€”';
    appendLog('UI cleared');
    simulateCaptureLevel(5);
  });

  // View DB - open a new window/tab with pretty JSON (client-only)
  viewDbBtn.addEventListener('click', ()=>{
    const key = dbSelect.value;
    const db = DATABASES[key];
    if(!db){ alert('DB not found.'); return; }
    const w = window.open('', '_blank');
    w.document.title = 'DB: ' + key;
    const pre = w.document.createElement('pre');
    pre.style.background = '#071022';
    pre.style.color = '#e6eef8';
    pre.style.padding = '14px';
    pre.style.borderRadius = '8px';
    pre.textContent = JSON.stringify(db, null, 2);
    w.document.body.style.background = '#02101a';
    w.document.body.style.margin = '20px';
    w.document.body.appendChild(pre);
    appendLog('Viewed DB: ' + key);
  });
```

```
  // Export DB JSON
  exportDbBtn.addEventListener('click', ()=>{
    const key = dbSelect.value;
    const db = DATABASES[key];
    if(!db){ alert('DB not found.'); return; }
    const blob = new Blob([JSON.stringify(db, null, 2)], {type: 'application/
json'});
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `${key}.json`;
    document.body.appendChild(a);
    a.click();
    a.remove();
    URL.revokeObjectURL(url);
    appendLog('Exported DB: ' + key);
  });

  // Import DB (file input)
  importDbBtn.addEventListener('click', ()=> dbFile.click());
  dbFile.onchange = (e)=>{
    const f = e.target.files[0];
    if(!f) return;
    const reader = new FileReader();
    reader.onload = (ev)=>{
      try {
        const parsed = JSON.parse(ev.target.result);
        const key = parsed.database_name || parsed.full_name || ('DB_' +
Date.now());
        DATABASES[key] = parsed;
        refreshDbList();
        dbSelect.value = key;
        appendLog('Imported DB: ' + key);
        alert('Imported DB as key: ' + key);
      } catch(err){
        alert('Invalid JSON file');
      }
    };
    reader.readAsText(f);
    dbFile.value = '';
  };

  // Add placeholder DB (simple empty template)
  addPlaceholderBtn.addEventListener('click', ()=>{
    const key = 'PLACEHOLDER_' + Math.floor(Math.random()*10000);
    const template = {
      database_name: key,
```

```
      full_name: 'Placeholder DB ' + key,
      version: '0.0.1',
      created_by: 'User',
      creation_date: new Date().toISOString().slice(0,10),
      purpose: 'Placeholder / future DB',
      notes: 'This is a user-created placeholder DB'
    };
    DATABASES[key] = template;
    refreshDbList();
    dbSelect.value = key;
    appendLog('Added placeholder DB: ' + key);
    alert('Placeholder DB added: ' + key);
  });

  // ===== OAX control functions (simulation-only) =====
  function logToOaxTerminal(message){
    const t = document.getElementById('oaxTerminal');
    const ts = new Date().toLocaleTimeString();
    t.innerHTML += `[${ts}] ${message}<br>`;
    t.scrollTop = t.scrollHeight;
    appendLog('OAX: ' + message);
  }

  function executeCreateCode(){
    const code =
document.getElementById('createCodeInput').value.trim();
    if (!code) {
      logToOaxTerminal('ERROR: No CREATE code provided');
      return;
    }
    if (code.startsWith('create.')) {
      logToOaxTerminal(`Executing: ${code}`);
      setTimeout(() => {
        logToOaxTerminal('Command processed - Non-electric systems
responding (simulation)');
        updateOAXStatus();
      }, 700);
    } else {
      logToOaxTerminal('ERROR: Invalid CREATE code format');
    }
  }

  function initializeBrainReader(){
    logToOaxTerminal('Initializing brain reader systems...');
    logToOaxTerminal('Loading atererean power source...');
    logToOaxTerminal('Calibrating OAX rotary position...');
    setTimeout(() => {
      logToOaxTerminal('Brain reader initialized - Ready for decoding
```

```
(simulation)');
      updateOAXStatus();
    }, 1200);
  }

  let isDecodingActive = false;
  function startDecoding(){
    if (!isDecodingActive) {
      isDecodingActive = true;
      logToOaxTerminal('Starting brain decoding process (simulation)...');
      logToOaxTerminal('Activating non- electric neural sensors
(simulated)...');
      logToOaxTerminal('Scanning for thought patterns...');
      setTimeout(() => {
        logToOaxTerminal('SIMULATION: Thought patterns detected
(fake)');
        logToOaxTerminal('WARNING: Actual brain reading not possible in
this UI');
      }, 900);
    } else {
      logToOaxTerminal('Decoding already active');
    }
  }

  function cloneBrain(){
    logToOaxTerminal('Initiating brain cloning sequence (simulation)...');
    setTimeout(() => {
      logToOaxTerminal('Brain clone created - sending to magnar storage
(simulation)');
      logToOaxTerminal('Original thoughts preserved (simulated
snapshot)');
    }, 1200);
  }

  function brainCommand(command) {
    const commands = {
      jump: 'create.asktojump.start',
      split: 'create.asktosplit.start',
      merge: 'create.asktomerge.start',
      reveal: 'create.asktoreveal.start',
      clone: 'create.startbraincloningorduplication.start',
      reset: 'create.brainreset086688xtu.start'
    };
    const cmd = commands[command] || 'create.unknown';
    const bt = document.getElementById('brainTerminal');
    const ts = new Date().toLocaleTimeString();
    bt.innerHTML += `[${ts}] Executing: ${cmd}< br>`;
    bt.scrollTop = bt.scrollHeight;
```

```
    appendLog('BrainCommand: ' + cmd);
    setTimeout(() => {
      bt.innerHTML += `[${new Date().toLocaleTimeString()}] Brain
command '${command}' processed<br>`;
      bt.scrollTop = bt.scrollHeight;
    }, 700);
  }

  function extractThoughts(){
    const file = document.getElementById('imageInput').files[0];
    const t = document.getElementById('thoughtOutput');
    const ts = new Date().toLocaleTimeString();
    if (file) {
      t.innerHTML += `[${ts}] Processing image: ${file.name}<br>`;
      t.scrollTop = t.scrollHeight;
      setTimeout(() => {
        t.innerHTML += `[${new Date().toLocaleTimeString()}] SIMULATION
RESULT: Thought extraction complete (simulated)<br>`;
        t.innerHTML += `[${new Date().toLocaleTimeString()}] ACTUAL
CAPABILITY: Cannot read thoughts from images<br>`;
        t.scrollTop = t.scrollHeight;
      }, 1100);
    }
  }

  function updateOAXStatus(){
    const angles = ['85Â° North', '72Â° Northeast', '90Â° East', '78Â°
Southeast'];
    document.getElementById('oax-position').textContent =
angles[Math.floor(Math.random() * angles.length)];
    document.getElementById('rotary-speed').textContent =
Math.floor(Math.random() * 200) + ' RPM';
    document.getElementById('long-ago').textContent = (8.0 +
Math.random() * 0.1).toFixed(2) + ' sec';
    document.getElementById('atererean').textContent = (0.869 +
Math.random()*0.002).toFixed(6);
  }

  // periodic OAX logs
  setInterval(() => {
    if (Math.random() < 0.12) {
      const updates = [
        'Neural field fluctuation detected (sim)',
        'OAX rotary calibration stable (sim)',
        'Non-electric power maintaining levels (sim)',
        'Thought pattern buffer ready (sim)'
      ];
      logToOaxTerminal(updates[Math.floor(Math.random() *
```

```
updates.length)]);
    }
  }, 5200);

  // Populate DB selector initial
  simulateCaptureLevel(6);
  appendLog('AANIC GUI loaded (TWODOC included as placeholder DB and OAX panel integrated).');
 </script>


s



 <meta name="viewport" content="width=device-width,initial-scale=1">
 <title>AANIC â€” Brain Codec Simulator (with TWODOC placeholder DB)</title>
 <style>
  :root{
    --bg:#0f1724; --card:#0b1220; --muted:#9aa7bf; --accent:#6ee7b7;
    --glass: rgba(255,255,255,0.03);
    font-family: Inter, system-ui, -apple-system, "Segoe UI", Roboto, "Helvetica Neue", Arial;
  }
  html,body{height:100%; margin:0; background:linear-gradient(180deg,#071024 0%, #071827 60%); color:#e6eef8}
  .wrap{max-width:1200px; margin:24px auto; display:grid; grid-template-columns:380px 1fr 380px; gap:18px; padding:18px;}
  .card{background:var(--card); padding:14px; border-radius:12px; box-shadow:0 6px 24px rgba(2,6,23,0.6); border:1px solid rgba(255,255,255,0.03)}
  h1{font-size:18px; margin:0 0 8px 0}
  .peripherals .row{display:flex;gap:8px;align-items:center;margin:8px 0}
  .btn{background:var(--accent); color:#012; border:none; padding:8px 12px;border-radius:8px; cursor:pointer}
  .toggle{display:inline-flex;align-items:center;gap:8px}
  .chat{display:flex;flex-direction:column;height:620px}
  .messages{flex:1; overflow:auto; padding:8px; border-radius:8px; background:linear-gradient(180deg, rgba(255,255,255,0.01), transparent)}
  .inputRow{display:flex; gap:8px; margin-top:8px}
  textarea{width:100%;min-height:56px;border-radius:8px;padding:10px;border:1px solid rgba(255,255,255,0.04); background:transparent;color:inherit}
  .small{font-size:13px; color:var(--muted)}
  .simBox{background:linear-gradient(180deg, rgba(255,255,255,0.02), transparent); padding:10px; border-radius:8px; font-family:monospace; max-height:260px; overflow:auto}
```

```
   .chip{display:inline-block;padding:6px 8px;border-
radius:999px;background:rgba(255,255,255,0.03); font-size:13px}
   footer{grid-column:1/-1; text-align:center; color:var(--muted);
padding-top:6px}
   .status{display:flex; gap:8px; align-items:center; margin-bottom:10px}
   .meter{height:8px;background:#071827;border-
radius:8px;overflow:hidden}
   .meter > i{display:block;height:100%; background:linear-
gradient(90deg,#00f 0%, #6ee7b7 100%)}
   .legend{display:flex;gap:8px;margin-top:6px;font-
size:12px;color:var(--muted)}
   .compare{display:grid; grid-template-columns:1fr 1fr; gap:8px}
   .note{font-
size:13px;color:#ffd6d6;background:rgba(255,50,50,0.05);padding:8px;bo
rder-radius:8px}
   .dbControls{display:flex;gap:8px;flex-wrap:wrap;margin-top:8px}
   .fileInput{display:none}
 </style>


 <div class="wrap">
  <!-- LEFT: Peripherals & DB -->
  <section class="card peripherals">
   <h1>AANIC â€” Peripheral &amp;amp; Database Panel</h1>
   <div class="small">TWODOC is loaded as a placeholder database
(creator-provided JSON). Use the DB tools to view/export/import future
databases.</div>

    <div style="margin-top:12px">
     <div class="row"><label class="chip">Database ID</label><input
id="DatabaseId" placeholder="Database82698" style="flex:1;border-
radius:8px;padding:6px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit"></div>

     <div class="row">
      <label class="toggle"><input type="checkbox" id="connectProp">
Rotary Propeller</label>
     </div>
     <div class="row">
      <label class="toggle"><input type="checkbox" id="connectDiode">
Needle Diode (CNB)</label>
     </div>
     <div class="row">
      <label class="toggle"><input type="checkbox" id="connectMirror">
Mirror-Image Sync</label>
     </div>

     <hr style="border:none;border-top:1px solid
```

```
rgba(255,255,255,0.03);margin:12px 0">
      <div><strong>Database</strong></div>
      <div style="margin-top:8px" class="row">
        <select id="dbSelect" style="flex:1;padding:8px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit">
          <!-- options dynamically populated -->
        </select>
      </div>

      <div class="dbControls">
        <button class="btn" id="viewDbBtn">View DB</button>
        <button class="btn" id="exportDbBtn"
style="background:#8ab4ff;color:#002">Export DB JSON</button>
        <button class="btn" id="importDbBtn"
style="background:transparent;color:var(--muted);border:1px solid
rgba(255,255,255,0.03)">Import DB</button>
        <input type="file" id="dbFile" class="fileInput"
accept=".json,application/json">
        <button class="btn" id="addPlaceholderBtn"
style="background:#ffd27a;color:#102">Add Placeholder DB</button>
      </div>

      <div style="margin-top:12px"><strong>Model</strong></div>
      <div style="margin-top:8px" class="row">
        <select id="modelSelect" style="flex:1;padding:8px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit">
          <option value="geto888g8678">geto888g8678 (geto/LLaMA-style,
simulated)</option>
          <option value="claude_like">Claude-like (simulated comparator)</
option>
        </select>
      </div>

      <div style="margin-top:12px" class="row">
        <button class="btn" id="genSignal">Generate Sim Signal</button>
        <button class="btn" id="trainSim"
style="background:#6ee7ff;color:#002">Run Training (sim)</button>
      </div>

      <div style="margin-top:12px" class="small">
        <div><strong>Training guidance (sim):</strong> simulation logs
DB usage and mapping. This remains a UI-only sandbox â€” no
hardware access or medical claims are implemented by the tool.</div>
      </div>
    </div>
  </section>
```

```
  <!-- CENTER: Chat & codec -->
  <main class="card chat">
    <h1>AANIC Chat â€” AANIC DAC (Digital Analogue Codec) Simulator</h1>
    <div class="status">
      <div class="small">Simulated capture level</div>
      <div class="meter" style="flex:1"><i id="meterBar" style="width:9%"></i></div>
      <div id="captureHint" class="small">idle</div>
    </div>

    <div class="messages" id="messages" aria-live="polite"></div>

    <div class="inputRow">
      <textarea id="userInput" placeholder="Type a prompt or &#39;decode signal&#39; to ask the codec..."></textarea>
    </div>
    <div style="display:flex;gap:8px;margin-top:8px">
      <button class="btn" id="sendBtn">Send â†’</button>
      <button id="decodeBtn" class="btn" style="background:#ffd27a;color:#102">Decode Signal</button>
      <button id="clearBtn" class="btn" style="background:transparent;color:var(--muted);border:1px solid rgba(255,255,255,0.03)">Clear</button>
    </div>

    <div style="margin-top:12px" class="small">NOTE: This demo decodes *simulated* signals via deterministic JS heuristics + the chosen mock DB. It is not a real brain reader; it provides a sandbox for UI and algorithm ideas only.</div>
  </main>

  <!-- RIGHT: Simulation panels + comparison -->
  <aside class="card">
    <h1>Sim Engine</h1>
    <div class="small">AANIC DAC maps simulated vibration patterns â†’ candidate phrases using loaded database contents (TWODOC loaded by default).</div>

    <div style="margin-top:10px">
      <div><strong>Last raw signal</strong></div>
      <div id="rawSignal" class="simBox">â€” none â€”</div>

      <div style="margin-top:12px"><strong>Decoded output</strong></div>
      <div id="decodedBox" class="simBox">â€” none â€”</div>
```

```
      <div style="margin-top:12px"><strong>Comparison</strong></
div>
      <div class="compare">
        <div>
          <div class="small">geto888g8678</div>
          <div id="llamaOut" class="simBox">â€”</div>
        </div>
        <div>
          <div class="small">Claude-like</div>
          <div id="claudeOut" class="simBox">â€”</div>
        </div>
      </div>

      <div style="margin-top:12px">
        <div class="small">Activity Log</div>
        <div id="log" class="simBox" style="height:120px"></div>
      </div>

      <div style="margin-top:12px" class="note">TWODOC JSON included
verbatim as a placeholder DB. This tool does not publish or transmit
data anywhere â€” it runs locally in your browser.</div>
    </div>
  </aside>

  <footer class="small">
    Prototype with placeholder DB support. To turn this into a research
pipeline you'd need approved neurotech sensors, ethics review, and
secure servers â€” this demo avoids any hardware instructions.
  </footer>
 </div>

 <script>
  // ------------------------
  // TWODOC placeholder (exact JSON provided by user)
  // ------------------------
  const TWODOC = {
 "database_name": "TWODOC",
 "full_name": "Tomorrow's World Order Doctor Database",
 "version": "888500.1.0",
 "classification": "888500 Critical Issues Resolved",
 "created_by": "Tomorrow's World Order",
 "creation_date": "2025-08-23",
 "purpose": "AI medical database for death reversal and immortality
protocols",
 "system_overview": {
  "total_issues": 500,
  "resolved_issues": 500,
  "success_rate": "100%",
```

```
    "primary_focus": "Complete reversal of aging and death prevention",
    "target_lifespan": "29 billion years",
    "core_technology": "AGT (Advanced Genetic Transformation)"
  },
  "cellular_molecular_biology": {
    "category_range": "Issues 1-100",
    "status": "RESOLVED",
    "key_protocols": {
      "cellular_senescence_reversal": {
        "method": "Complete reversal mechanism using AGT codes",
        "effectiveness": "100% reversal from aged to youthful state",
        "implementation": "Create codes with x-removal process"
      },
      "telomere_maintenance": {
        "current_length": "72cm (enhanced from 8cm)",
        "protocol": "AGT-based extension using 8000 AGT units",
        "longevity_impact": "Indefinite cellular division capability"
      },
      "dna_repair_system": {
        "sequence_count": "53.8 billion DNA sequences",
        "repair_capacity": "Daily damage detection and protein misfolding
prevention",
        "comparison": "Human baseline: 72 million sequences"
      },
      "mitochondrial_optimization": {
        "dysfunction_elimination": "Complete restoration protocol",
        "energy_production": "Optimized ATP generation"
      },
      "stem_cell_regeneration": {
        "exhaustion_reversal": "AGT-based stem cell renewal",
        "implementation": "Continuous cellular replacement"
      }
    }
  },
  "organ_system_maintenance": {
    "category_range": "Issues 101-200",
    "status": "RESOLVED",
    "cardiovascular_systems": {
      "endothelial_function": "Preserved indefinitely",
      "cardiac_regeneration": "Enhanced muscle regeneration capability",
      "blood_pressure": "Perfect regulation mechanisms",
      "circulation": "Optimized micro and macro circulation"
    },
    "respiratory_systems": {
      "epithelium_regeneration": "X-Y-E regeneration protocol
implemented",
      "gas_exchange": "Optimized oxygen carrying capacity",
      "lung_compliance": "Maintained for eternal function"
```

```
  },
  "renal_systems": {
    "glomerular_function": "Preserved filtration capabilities",
    "electrolyte_balance": "Automated regulation systems",
    "detoxification": "Enhanced clearance mechanisms"
  },
  "hepatic_systems": {
    "detoxification_enhancement": "Complete toxin elimination",
    "protein_synthesis": "Optimized albumin and factor production",
    "metabolic_regulation": "Perfect glucose and lipid metabolism"
  }
 },
 "neurological_cognitive_preservation": {
  "category_range": "Issues 201-250",
  "status": "RESOLVED",
  "neural_optimization": {
    "membrane_integrity": "Maintained indefinitely",
    "synaptic_transmission": "Enhanced efficiency",
    "neurotransmitter_balance": "Perfect regulation of all systems"
  },
  "cognitive_enhancement": {
    "memory_consolidation": "Optimized long-term potentiation",
    "neuroplasticity": "Maintained synaptic adaptability",
    "neurotrophin_systems": "Enhanced BDNF and NGF production"
  }
 },
 "sensory_system_preservation": {
  "category_range": "Issues 251-300",
  "status": "RESOLVED",
  "visual_systems": {
    "retinal_maintenance": "Photoreceptor preservation protocols",
    "lens_clarity": "Indefinite transparency maintenance",
    "visual_acuity": "Optimized processing"
  },
  "auditory_systems": {
    "cochlear_preservation": "Hair cell maintenance protocols",
    "frequency_discrimination": "Enhanced hearing capabilities"
  },
  "other_senses": {
    "olfactory_maintenance": "Receptor neuron preservation",
    "tactile_optimization": "Enhanced mechanoreceptor function"
  }
 },
 "musculoskeletal_maintenance": {
  "category_range": "Issues 301-350",
  "status": "RESOLVED",
  "muscle_systems": {
    "fiber_preservation": "Type I and II optimization",
```

```
    "protein_synthesis": "Enhanced muscle maintenance",
    "strength_maintenance": "Indefinite power output"
   },
   "skeletal_systems": {
    "bone_density": "Optimal calcium phosphate deposition",
    "remodeling_balance": "Perfect osteoblast/osteoclast regulation",
    "cartilage_preservation": "Chondrocyte function optimization"
   }
  },
  "endocrine_optimization": {
   "category_range": "Issues 351-400",
   "status": "RESOLVED",
   "hormonal_systems": {
    "hypothalamic_pituitary": "Perfect axis maintenance",
    "thyroid_optimization": "T3/T4 balance maintenance",
    "adrenal_function": "Optimal cortisol rhythm",
    "reproductive_hormones": "Balanced sex hormone production",
    "metabolic_hormones": "Insulin sensitivity preservation"
   }
  },
  "safety_toxicology": {
   "category_range": "Issues 401-450",
   "status": "RESOLVED",
   "toxicity_elimination": {
    "acute_toxicity": "Complete absence achieved",
    "chronic_toxicity": "Total elimination protocols",
    "organ_specific_toxicity": "Prevention across all systems",
    "carcinogenicity": "Absolute elimination",
    "mutagenicity": "Complete absence maintained"
   },
   "interaction_safety": {
    "drug_interactions": "Complete safety profile",
    "environmental_safety": "No bioaccumulation or biomagnification",
    "demographic_safety": "Safe across all populations"
   }
  },
  "implementation_deployment": {
   "category_range": "Issues 451-500",
   "status": "RESOLVED",
   "manufacturing": {
    "scalability": "Proven global production capability",
    "quality_control": "GMP compliance achieved",
    "supply_chain": "Secured global distribution"
   },
   "regulatory_approval": {
    "fda_pathway": "Navigation protocols established",
    "international_harmonization": "WHO prequalification pursued",
    "patent_protection": "Intellectual property secured"
```

```
    },
    "global_access": {
      "healthcare_education": "Provider training programs",
      "patient_education": "Comprehensive material development",
      "equity_considerations": "Developing nation access programs"
    }
  },
  "technical_specifications": {
    "agt_system": {
      "units_required": "8000 AGT units for full implementation",
      "license_code": "Creator license:
ajertert63ertertuert76283888888888888888888888",
      "implementation_age": "Optimal at age 49- 50 for prevention
protocols"
    },
    "monitoring_systems": {
      "biomarkers": "Comprehensive tracking of all 500 parameters",
      "adverse_events": "Real- time monitoring protocols",
      "efficacy_metrics": "Continuous lifespan extension validation"
    }
  },
  "research_validation": {
    "animal_models": {
      "tortoises": "Longevity mechanisms studied and replicated",
      "lobsters": "Indefinite growth patterns incorporated",
      "greenland_sharks": "Daily regeneration systems adapted"
    },
    "human_trials": {
      "subject": "David Gomadza -  Age reverted to 12, potential 29 billion
year lifespan",
      "biomarkers": "All 500 critical issues successfully resolved",
      "side_effects": "Zero adverse events recorded"
    }
  },
  "database_access": {
    "authorization_level": "Creator Only",
    "access_codes": "Secured with AGT authentication",
    "update_frequency": "Real- time continuous monitoring",
    "backup_systems": "Multi- dimensional storage across time streams"
  },
  "contact_information": {
    "organization": "Tomorrow's World Order",
    "president": "David Gomadza",
    "email": "davidgomadza@hotmail.com",
    "website": "www.twofuture.world",
    "phone": "00447719210295",
    "status": "President of the World"
  },
```

```
  "database_validation": {
    "checksum": "888500.verification.complete",
    "integrity_status": "VALIDATED",
    "last_updated": "2025-06-20",
    "next_review": "Continuous monitoring active",
    "certification": "Complete cure for death -  29 billion years validated"
  }
};

    // ------------------------
    // Runtime DB store (in-memory)
    // ------------------------
    const DATABASES = {
      '82698': { database_name: '82698', description: 'Mock Subject Core DB
(sim)' },
      'brain_codes': { database_name: 'brain_codes', description: 'Mock
Brain Codes DB (sim)' },
      'TWODOC': TWODOC
    };

    // ------------------------
    // UI refs
    // ------------------------
    const dbSelect = document.getElementById('dbSelect');
    const viewDbBtn = document.getElementById('viewDbBtn');
    const exportDbBtn = document.getElementById('exportDbBtn');
    const importDbBtn = document.getElementById('importDbBtn');
    const dbFile = document.getElementById('dbFile');
    const addPlaceholderBtn =
document.getElementById('addPlaceholderBtn');

    // messages / sim refs
    const messages = document.getElementById('messages');
    const rawSignal = document.getElementById('rawSignal');
    const decodedBox = document.getElementById('decodedBox');
    const llamaOut = document.getElementById('llamaOut');
    const claudeOut = document.getElementById('claudeOut');
    const log = document.getElementById('log');
    const meterBar = document.getElementById('meterBar');
    const captureHint = document.getElementById('captureHint');

    // ------------------------
    // Populate DB selector
    // ------------------------
    function refreshDbList(){
      dbSelect.innerHTML = '';
      Object.keys(DATABASES).forEach(k=>{
        const opt = document.createElement('option');
```

```
      opt.value = k;
      opt.textContent = `${k} â€” ${DATABASES[k].database_name ||
DATABASES[k].full_name || 'Unnamed DB'}`;
      dbSelect.appendChild(opt);
    });
  }
  refreshDbList();

  // ------------------------
  // Logging & UI helpers
  // ------------------------
  function pushMsg(role, text){
    const el = document.createElement('div');
    el.innerHTML = `<div style="margin:8px 0"><strong
style="color:#9dd3c5">${role}</strong><div style="margin-top:6px">$
{escapeHtml(text)}</div></div>`;
    messages.appendChild(el);
    messages.scrollTop = messages.scrollHeight;
  }
  function appendLog(t){ log.textContent = (new
Date()).toLocaleTimeString() + ' â€” ' + t + '\n' + log.textContent; }
  function escapeHtml(s){ return (''+s).replace(/&/g,'&amp;').replace(/</
g,'&lt;').replace(/>/g,'&gt;').replace(/\\n/g,'<br>'); }

  // Fake brain-signal generator (random patterns)
  function generateSimSignal(){
    const patterns = [
      {code:'A-7-101',desc:'visual attention: screen/monitor',phrases:
['looking at screen','checking messages','reading a webpage']},
      {code:'B-3-522',desc:'hunger or food thought',phrases:['I want
food','hungry','thinking about dinner']},
      {code:'C-9-007',desc:'familiar voice memory',phrases:['I hear a
voice','my name was called','someone said hello']},
      {code:'D-2-880',desc:'movement intent',phrases:['I will stand
up','reach for phone','move left']},
      {code:'E-6-82698',desc:'subject-core code (DB 82698
marker)',phrases:['twofuture.world','david gomadza','brain code demo']}
    ];
    const p = patterns[Math.floor(Math.random()*patterns.length)];
    return {timestamp:Date.now(), code:p.code, desc:p.desc,
phrases:p.phrases};
  }

  function simulateCaptureLevel(intensity){
    meterBar.style.width = Math.max(6, Math.min(100, intensity)) + '%';
    captureHint.textContent = intensity > 60 ? 'strong' : intensity > 30 ?
'moderate' : 'weak';
  }
```

```
  // New: codec uses DB content to bias outputs
  function runAanicCodec(signal, dbKey){
    const db = DATABASES[dbKey] || {};
    let out = '';
    // Priority rules:
    // 1) If DB has 'database_name' TWODOC and 'purpose' includes
'death' => produce a phrase referencing lifespan/death reversal
    // 2) If code contains 82698 or dbKey is 82698, pick subject-core
phrases
    // 3) Fallback: map pattern -> phrase
    try {
      if(db && db.purpose && /death|immortality|lifespan|reversal/
i.test(db.purpose)){
        // craft an output that references DB context (simulation)
        out = `DB:${db.database_name || db.database_name} context â†’ "$
{signal.phrases[0]}" ; DB-purpose: ${db.purpose}. Note: $
{db.system_overview ? db.system_overview.primary_focus : 'primary
focus unknown'}.`;
      } else if(signal.code.includes('82698') || dbKey==='82698'){
        out = `Subject core hit â†’ probable reference: "$
{signal.phrases[0]}" (from DB ${dbKey})`;
      } else {
        const phrase =
signal.phrases[Math.floor(Math.random()*signal.phrases.length)];
        out = `Decoded pattern ${signal.code} (${signal.desc}) â†’ "$
{phrase}"`;
      }
    } catch(e){
      out = `Decoded pattern ${signal.code} (${signal.desc}) â†’ "$
{signal.phrases[0]}"`;
    }
    return out;
  }

  function mockModelResponse(decodedText, model){
    if(model === 'geto888g8678'){
      return `LLAMA-sim: ${decodedText.replace('â†',|')}\nConfidence: $
{Math.floor(Math.random()*15)+82}%`;
    } else {
      return `CLAUDE-sim: Interpreting the decoded pattern, this
suggests: ${decodedText.split('â†').pop().trim()}. Possible contexts:
memory, attention, or intent. Confidence ~ $
{Math.floor(Math.random()*12)+78}%`;
    }
  }

  // UI button bindings
```

```
  document.getElementById('genSignal').onclick = ()=>{
    const sig = generateSimSignal();
    rawSignal.textContent = JSON.stringify(sig, null, 2);
    appendLog('Generated sim signal: ' + sig.code + ' â€” ' + sig.desc);
    simulateCaptureLevel(Math.floor(Math.random()*80)+10);
    pushMsg('SYSTEM', `Simulated raw capture: ${sig.code} â€” ${sig.desc}
`);
    // auto-decode preview using selected DB
    const dbKey = dbSelect.value;
    const decoded = runAanicCodec(sig, dbKey);
    decodedBox.textContent = decoded;
    appendLog('AANIC DAC decoded: ' + decoded);
    // model outputs
    llamaOut.textContent = mockModelResponse(decoded,
'geto888g8678');
    claudeOut.textContent = mockModelResponse(decoded,
'claude_like');
  };

  document.getElementById('trainSim').onclick = ()=>{
    appendLog('Simulated training pass executed (4 wks equiv
simulated). DB used: ' + dbSelect.value);
    pushMsg('TRAIN', 'Simulated model training run complete. (UI-only.)');
  };

  document.getElementById('sendBtn').onclick = ()=>{
    const txt = document.getElementById('userInput').value.trim();
    if(!txt) return;
    pushMsg('You', txt);
    document.getElementById('userInput').value = '';
    appendLog('User prompt: ' + txt);
    if(/decode|signal|brain/i.test(txt)){
      document.getElementById('genSignal').click();
      pushMsg('AANIC', 'Decoding latest simulated signal and returning
candidate phrases (simulation only).');
    } else {
      const model = document.getElementById('modelSelect').value;
      const decoded =
(document.getElementById('decodedBox').textContent || 'â€” none â€”');
      const response = mockModelResponse(decoded === 'â€” none
â€”' ? txt : decoded, model);
      pushMsg('AANIC', response);
    }
  };

  document.getElementById('decodeBtn').onclick = ()=>{
    const raw = rawSignal.textContent;
    if(!raw || raw.trim()==='â€” none â€—'){
```

```
      pushMsg('AANIC', 'No raw simulated signal present. Click "Generate
Sim Signal" first.');
      return;
    }
    const dbKey = dbSelect.value;
    const parsed = JSON.parse(rawSignal.textContent);
    const decoded = runAanicCodec(parsed, dbKey);
    decodedBox.textContent = decoded;
    pushMsg('AANIC', 'AANIC DAC decoded (sim): ' + decoded);
    llamaOut.textContent = mockModelResponse(decoded,
'geto888g8678');
    claudeOut.textContent = mockModelResponse(decoded,
'claude_like');
    appendLog('Manual decode executed. DB: ' + dbKey);
  };

  document.getElementById('clearBtn').onclick = ()=>{
    messages.innerHTML=''; rawSignal.textContent='â€” none â€”';
decodedBox.textContent='â€” none â€”'; llamaOut.textContent='â€”';
claudeOut.textContent='â€”';
    appendLog('UI cleared');
    simulateCaptureLevel(5);
  };

  // View DB - open a new window/tab with pretty JSON (client-only)
  viewDbBtn.onclick = ()=>{
    const key = dbSelect.value;
    const db = DATABASES[key];
    if(!db){ alert('DB not found.'); return; }
    const w = window.open('', '_blank');
    w.document.title = 'DB: ' + key;
    const pre = w.document.createElement('pre');
    pre.style.background = '#071022';
    pre.style.color = '#e6eef8';
    pre.style.padding = '14px';
    pre.style.borderRadius = '8px';
    pre.textContent = JSON.stringify(db, null, 2);
    w.document.body.style.background = '#02101a';
    w.document.body.style.margin = '20px';
    w.document.body.appendChild(pre);
    appendLog('Viewed DB: ' + key);
  };

  // Export DB JSON
  exportDbBtn.onclick = ()=>{
    const key = dbSelect.value;
    const db = DATABASES[key];
    if(!db){ alert('DB not found.'); return; }
```

```
    const blob = new Blob([JSON.stringify(db, null, 2)], {type: 'application/
json'});
    const url = URL.createObjectURL(blob);
    const a = document.createElement('a');
    a.href = url;
    a.download = `${key}.json`;
    document.body.appendChild(a);
    a.click();
    a.remove();
    URL.revokeObjectURL(url);
    appendLog('Exported DB: ' + key);
  };

  // Import DB (file input)
  importDbBtn.onclick = ()=> dbFile.click();
  dbFile.onchange = (e)=>{
   const f = e.target.files[0];
   if(!f) return;
   const reader = new FileReader();
   reader.onload = (ev)=>{
    try {
     const parsed = JSON.parse(ev.target.result);
     const key = parsed.database_name || parsed.full_name || ('DB_' +
Date.now());
     DATABASES[key] = parsed;
     refreshDbList();
     dbSelect.value = key;
     appendLog('Imported DB: ' + key);
     alert('Imported DB as key: ' + key);
    } catch(err){
     alert('Invalid JSON file');
    }
   };
   reader.readAsText(f);
   dbFile.value = '';
  };

  // Add placeholder DB (simple empty template)
  addPlaceholderBtn.onclick = ()=>{
   const key = 'PLACEHOLDER_' + Math.floor(Math.random()*10000);
   const template = {
    database_name: key,
    full_name: 'Placeholder DB ' + key,
    version: '0.0.1',
    created_by: 'User',
    creation_date: new Date().toISOString().slice(0,10),
    purpose: 'Placeholder / future DB',
    notes: 'This is a user- created placeholder DB'
```

```
    };
    DATABASES[key] = template;
    refreshDbList();
    dbSelect.value = key;
    appendLog('Added placeholder DB: ' + key);
    alert('Placeholder DB added: ' + key);
  };

  // init
  simulateCaptureLevel(6);
  appendLog('AANIC GUI loaded (TWODOC included as placeholder
DB).');

 </script>


s



<meta name="viewport" content="width=device-width,initial-scale=1">
<title>Adaptive Brain AI â€” Live API Integration</title>
<style>
 :root{--bg:#0f1724;--card:#111420;--accent:#00d1ff;--muted:#94a3b8}
 body{font-family:Inter,Segoe UI,Arial;background:linear-
gradient(180deg,#071023 0%, #071028
100%);color:#e6eef6;margin:0;padding:24px}
 .wrap{max-width:1000px;margin:0 auto;background:var(--
card);padding:20px;border-radius:12px;box-shadow:0 10px 30px
rgba(2,6,23,0.6)}
 header{display:flex;align-items:center;gap:16px}
 header h1{margin:0;color:var(--accent);font-size:20px}
 .controls{display:flex;gap:8px;margin-top:14px}
 input[type=text]{flex:1;padding:10px;border-radius:8px;border:1px
solid rgba(255,255,255,0.05);background:transparent;color:inherit}
 button{background:var(--
accent);color:#001;border:none;padding:10px 14px;border-
radius:8px;cursor:pointer;font-weight:600}
 .grid{display:grid;grid-template-columns:1fr 360px;gap:16px;margin-
top:18px}
 .panel{background:#0b1220;padding:12px;border-
radius:10px;border:1px solid rgba(255,255,255,0.02)}
 #output{height:420px;overflow:auto;padding:8px;font-family:ui-
monospace, SFMono-Regular, Menlo, Monaco, monospace;white-
space:pre-wrap}
 .tasks{display:flex;flex-wrap:wrap;gap:8px;margin-top:10px}
 .task{background:transparent;border:1px solid
rgba(255,255,255,0.04);padding:8px 10px;border-
```

```
radius:8px;cursor:pointer}
 .dac{margin-top:12px}
 canvas{width:100%;height:140px;border-radius:8px;background:linear-
gradient(180deg, rgba(0,0,0,0.15), rgba(255,255,255,0.02));display:block}
 .small{font-size:12px;color:var(--muted)}
 footer{margin-top:12px;font-size:12px;color:var(--muted)}
</style>


 <div class="wrap">
  <header>
   <h1>AugVT â€” Adaptive Brain AI (Live API Ready)</h1>
   <div class="small">AANIC DAC â€¢ 7 Forms â€¢ DB 82698 â€¢
LLaMA â€¢ Claude</div>
  </header>

  <div class="controls">
   <input id="userInput" type="text" placeholder="Type or simulate your
thoughts...">
   <button id="decodeBtn">Decode</button>
   <button id="sendClaude">Send â†’ Claude</button>
   <button id="sendLlama">Send â†’ LLaMA</button>
  </div>

  <div class="grid">
   <div class="panel">
    <div class="tasks">
     <div class="task" data-task="summarize">Summarize</div>
     <div class="task" data-task="translate">Translate</div>
     <div class="task" data-task="emotion">Emotion</div>
     <div class="task" data-task="seven">7 Forms</div>
     <div class="task" data-task="db">Query DB 82698</div>
    </div>

    <div id="output"></div>

    <div class="small" style="margin-top:8px">Session Memory: <span
id="memCount">0</span> entries</div>
   </div>

   <div class="panel">
    <div class="dac">
     <div class="small">AANIC DAC â€” Neural Waveform Visualizer</
div>
     <canvas id="dacCanvas" width="800" height="140"></canvas>
     <div class="small" style="margin-top:8px">DAC Signal (hex
sample): <span id="dacHex">--</span></div>
    </div>
```

```
      <div style="margin-top:12px">
        <div class="small">Live API status:</div>
        <div id="status" class="small">Claude: <span
id="claudeStat">unknown</span> | LLaMA: <span
id="llamaStat">unknown</span> | DB: <span id="dbStat">unknown</
span></div>
      </div>
    </div>
  </div>

  <footer>
    Notes: To enable live AI, run a backend that proxies requests to your
AI providers. See top of file for sample Node/Express snippet. Do not
expose API keys in this front-end.
  </footer>
 </div>

<script>
// --- Frontend runtime logic ---
const sessionMemory = [];
const outputEl = document.getElementById('output');
const inputEl = document.getElementById('userInput');
const memCountEl = document.getElementById('memCount');
const dacHexEl = document.getElementById('dacHex');

// DAC canvas setup
const canvas = document.getElementById('dacCanvas');
const ctx = canvas.getContext('2d');
let width = canvas.width; let height = canvas.height;

function drawWave(samples){
 ctx.clearRect(0,0,width,height);
 ctx.beginPath();
 const step = width / samples.length;
 for(let i=0;i<samples.length;i++){
  const x = i*step;
  const v = (samples[i]/255)*height;
  const y = height - v;
  if(i===0) ctx.moveTo(x,y); else ctx.lineTo(x,y);
 }
 ctx.strokeStyle = '#00d1ff'; ctx.lineWidth = 2; ctx.stroke();
}

function simulateDAC(input){
 // create pseudo-random waveform from input
 const bytes = input.split('').map(c=>c.charCodeAt(0)%256);
 const memFactor = sessionMemory.length % 32;
```

```
  const samples = new Array(200).fill(0).map((_,i)=>{
    const idx = i % Math.max(1,bytes.length);
    return (bytes.length?((bytes[idx]+memFactor*(i%3))%256):
((i*7+memFactor)%256));
  });
  drawWave(samples);
  dacHexEl.textContent =
samples.slice(0,8).map(n=>n.toString(16).padStart(2,'0')).join(' ');
  return samples;
}

function appendOutput(text){
  const time = new Date().toLocaleTimeString();
  outputEl.textContent = `[${time}] ${text}\n\n` + outputEl.textContent;
}

// --- Tasks ---
function summarize(text){ return text.length>120? text.slice(0,120)+'...':
text; }
function translate(text){ return text.split('').reverse().join(''); }
function emotion(text){ const
moods=['Calm','Happy','Anxious','Sad','Excited','Neutral']; return
moods[(text.length+sessionMemory.length)%moods.length]; }
function sevenForms(text){ const
forms=['Text','Voice','Gesture','Visual','Emotion','Signal','Neural']; return
forms.map((f,i)=>`${f}: ${text.slice(0,i+1)}`).join('\n'); }

// --- Button events ---
document.getElementById('decodeBtn').addEventListener('click', async
()=>{
  const val = inputEl.value.trim(); if(!val) { appendOutput('Please type
thoughts to decode.'); return; }
  sessionMemory.push(val); memCountEl.textContent =
sessionMemory.length;
  simulateDAC(val);
  appendOutput('Adaptive Decode:\n'+
    'Input: '+val+'\n'+
    'Summary: '+summarize(val)+'\n'+
    'Emotion: '+emotion(val)+'\n'+
    '7- Forms:\n'+sevenForms(val)
  );
});

// Quick tasks
document.querySelectorAll('.task').forEach(btn=>
btn.addEventListener('click', ()=>{
  const task = btn.dataset.task; const val = inputEl.value.trim(); if(!val)
{ appendOutput('Type thoughts first.'); return; }
```

```
  sessionMemory.push(val);
memCountEl.textContent=sessionMemory.length; simulateDAC(val);
  let out='';
  if(task==='summarize') out='Summary: '+summarize(val);
  if(task==='translate') out='Translate (sim): '+translate(val);
  if(task==='emotion') out='Emotion: '+emotion(val);
  if(task==='seven') out='7- Forms:\n'+sevenForms(val);
  if(task==='db') callDB(val).then(r=> appendOutput('DB 82698 result:
\n'+JSON.stringify(r))).catch(e=> appendOutput('DB error: '+e.message));
  if(out) appendOutput(out);
}));

// --- Live API integration helpers ---
async function callClaude(input){
  // POST to backend /api/claude -> { input } returns { text }
  try{
    const res = await fetch('/api/claude',{method:'POST',headers:{'Content-
Type':'application/json'},body:JSON.stringify({input})});
    if(!res.ok) throw new Error('Network');
    const data = await res.json();
    return data.text || '[no text]';
  }catch(e){ return '[Claude fallback simulated] '+input; }
}
async function callLlama(input){
  try{
    const res = await fetch('/api/llama',{method:'POST',headers:{'Content-
Type':'application/json'},body:JSON.stringify({input})});
    if(!res.ok) throw new Error('Network');
    const data = await res.json();
    return data.text || '[no text]';
  }catch(e){ return '[LLaMA fallback simulated] '+input; }
}
async function callDB(query){
  try{
    const res = await fetch('/api/db82698',{method:'POST',headers:
{'Content- Type':'application/json'},body:JSON.stringify({query})});
    if(!res.ok) throw new Error('Network');
    const data = await res.json();
    return data.result || data;
  }catch(e){ return {error:'DB fallback', query}; }
}

// send to Claude
document.getElementById('sendClaude').addEventListener('click', async
()=>{
  const val = inputEl.value.trim(); if(!val) { appendOutput('Type thoughts
first.'); return; }
  appendOutput('Sending to Claude...');
```

```
  const reply = await callClaude(val);
  appendOutput('Claude â†’ '+reply);
});

// send to LLaMA
document.getElementById('sendLlama').addEventListener('click', async
()=>{
  const val = inputEl.value.trim(); if(!val) { appendOutput('Type thoughts
first.'); return; }
  appendOutput('Sending to LLaMA...');
  const reply = await callLlama(val);
  appendOutput('LLaMA â†’ '+reply);
});

// Check backend status (simple ping)
async function checkStatus(){
  try{ const c = await fetch('/api/claude',{method:'OPTIONS'});
document.getElementById('claudeStat').textContent = c.ok ? 'online' :
'unknown'; }catch(e)
{ document.getElementById('claudeStat').textContent='offline'; }
  try{ const l = await fetch('/api/llama',{method:'OPTIONS'});
document.getElementById('llamaStat').textContent = l.ok ? 'online' :
'unknown'; }catch(e)
{ document.getElementById('llamaStat').textContent='offline'; }
  try{ const d = await fetch('/api/db82698',{method:'OPTIONS'});
document.getElementById('dbStat').textContent = d.ok ? 'online' :
'unknown'; }catch(e)
{ document.getElementById('dbStat').textContent='offline'; }
}
checkStatus();

// Responsive canvas sizing
function resizeCanvas(){ width = canvas.width = canvas.clientWidth;
height = canvas.height = canvas.clientHeight; }
window.addEventListener('resize', ()=>{ resizeCanvas(); drawWave([]); });
resizeCanvas();

</script>


 s



<meta name="viewport" content="width=device- width,initial- scale=1">
<title> AANIC â€ ” Brain Thoughts â†’ Word &amp;amp; Audio (Enhanced
&amp;amp; Fixed)</title>
<style>
```

```
:root{--bg:#071024;--card:#071827;--accent:#7c3aed;--muted:#94a3b8;--good:#16a34a}
*{box-sizing:border-box;font-family:Inter,Segoe UI,system-ui,Arial,sans-serif}
body{background:linear-gradient(180deg,#041226,#071024);color:#e6eef8;margin:0;padding:20px}
.container{max-width:1100px;margin:0 auto}
.header{display:flex;gap:12px;align-items:center;justify-content:space-between}
.title{font-size:1.1rem;font-weight:700}
.grid{display:grid;grid-template-columns:1fr 420px;gap:18px;margin-top:18px}
.card{background:linear-gradient(180deg,rgba(255,255,255,0.02),rgba(255,255,255,0.01));padding:14px;border-radius:10px;border:1px solid rgba(255,255,255,0.03)}
.row{display:flex;gap:10px;align-items:center}
textarea{width:100%;height:100px;padding:10px;border-radius:8px;border:1px solid rgba(255,255,255,0.06);background:transparent;color:inherit}
select,input[type=range]{width:100%}
.btn{background:var(--accent);color:white;padding:9px 12px;border-radius:8px;border:none;cursor:pointer}
.btn.secondary{background:transparent;border:1px solid rgba(255,255,255,0.06)}
.indicator{width:12px;height:12px;border-radius:999px;background:#e74c3c}
.indicator.on{background:var(--good)}
.output{white-space:pre-wrap;background:rgba(0,0,0,0.28);padding:12px;border-radius:6px;font-family:ui-monospace,Menlo,monospace}
.canvas-wrap{height:120px;background:#041827;border-radius:6px;padding:6px}
.controls{display:flex;gap:8px;flex-wrap:wrap}
.dragdrop{border:2px dashed rgba(255,255,255,0.04);padding:12px;border-radius:8px;text-align:center;color:var(--muted);cursor:pointer}
.thumb{width:100%;height:80px;background:#03111a;border-radius:6px;margin-top:8px}
.footer{margin-top:18px;color:var(--muted);font-size:0.85rem}
.small{font-size:0.85rem;color:var(--muted)}
@media (max-width:900px){.grid{grid-template-columns:1fr}}
</style>


<div class="container">
  <div class="header">
    <div>
```

```
    <div class="title">AANIC â€” Brain Thoughts â†’ Word &amp;amp; Audio (Enhanced &amp;amp; Fixed)</div>
    <div class="small">Database 82698 â€¢ David Gomadza â€” DSP, drag &amp;amp; drop, waveform &amp;amp; spectrogram, segmentation</div>
   </div>
   <div style="display:flex;gap:10px;align-items:center">
    <div id="analogueIndicator" class="indicator"></div>
    <div id="readerIndicator" class="indicator"></div>
    <div id="statusLabel" class="small">Analogue: Disconnected Â· Reader: Offline</div>
   </div>
  </div>

  <div class="grid">
   <div class="card">
    <label><strong>EM pattern (worts / shorthand)</strong></label>
    <textarea id="input" placeholder="e.g. ikssyrghtnw">ikssyrghtnw</textarea>

    <div style="margin-top:10px" class="row">
     <div style="flex:1">
      <label>Thought type</label>
      <select id="thoughtType"><option>speech</option><option>emotion</option><option>command</option></select>
     </div>
     <div style="width:150px">
      <label>Resonance (Hz)</label>
      <input id="resFreq" type="range" min="1" max="100" value="50">
      <div id="resLabel" class="small">50 Hz</div>
     </div>
    </div>

    <div style="margin-top:12px;display:flex;gap:8px">
     <button id="convert" class="btn">Convert â†’</button>
     <button id="toggleReader" class="btn secondary">Start Real-Time</button>
     <button id="attachAnalogue" class="btn secondary">Attach Digital Analogue</button>
    </div>

    <div style="margin-top:12px">
     <div id="dragDrop" class="dragdrop">Drag &amp;amp; drop an audio file here, or click Attach Digital Analogue</div>
     <input id="hiddenFile" type="file" accept="audio/*" style="display:none">
     <div class="thumb"><canvas id="waveThumb" width="800" height="80"></canvas></div>
```

```
      <div class="thumb"><canvas id="specThumb" width="800"
height="80"></canvas></div>
    </div>

    <div style="margin-top:12px">
      <div><strong>Conversion output</strong></div>
      <div id="output" class="output">No conversion yet.</div>
    </div>

  </div>

  <div class="card">
    <div><strong>Audio &amp;amp; Neural Codec</strong></div>
    <div style="margin-top:8px">
      <label>Voice type</label>
      <select id="voiceSelect"><option value="neural">Neural</
option><option value="human">Human-like</option><option
value="robotic">Robotic</option></select>
    </div>

    <div style="margin-top:8px">
      <label>Base frequency</label>
      <input id="audioFreq" type="range" min="120" max="2000"
value="440">
      <div id="audioFreqLabel" class="small">440 Hz</div>
    </div>

    <div style="margin-top:12px" class="controls">
      <button id="genAudio" class="btn">Generate Audio</button>
      <button id="play" class="btn secondary">Play</button>
      <button id="stop" class="btn secondary">Stop</button>
      <button id="download" class="btn secondary">Download WAV</
button>
    </div>

    <div style="margin-top:12px">
      <div class="small"><strong>Segmentation &amp;amp; Matches</
strong></div>
      <div id="segments" class="small">(segments will appear here)</div>
    </div>

    <div style="margin-top:12px">
      <div class="small"><strong>DSP Match Scores</strong></div>
      <div id="matches" class="small">(matching info)</div>
    </div>

    <div style="margin-top:12px">
      <div style="font-size:0.9rem;color:var(--muted)"><strong>Status</
```

```
strong></div>
      <div id="audioStatus" class="small">No audio</div>
    </div>
  </div>
 </div>

 <div class="footer">This demo uses in-browser DSP: energy envelope,
cross-correlation, segmentation and visual FFT for spectrogram. It's for
experimentation only â€” real neural capture needs medical hardware.</
div>
</div>

<script>
// -------------------- Utilities & DB --------------------
const db = {
 ikssyrghtnw: {text: 'I kiss you right now', pattern:[220,330,440,330,220]},
 hlwdym: {text: 'Hello how are you my friend', pattern:
[330,440,550,440,330]},
 lvv: {text: 'love', pattern:[330,440,220]},
 kks: {text: 'kiss', pattern:[440,330,220]},
 jmp: {text: 'jump', pattern:[440,550,330]}
};

// DOM refs
const input = document.getElementById('input');
const convertBtn = document.getElementById('convert');
const output = document.getElementById('output');
const toggleReader = document.getElementById('toggleReader');
const attachAnalogue = document.getElementById('attachAnalogue');
const analogueIndicator =
document.getElementById('analogueIndicator');
const readerIndicator = document.getElementById('readerIndicator');
const statusLabel = document.getElementById('statusLabel');
const resFreq = document.getElementById('resFreq');
const resLabel = document.getElementById('resLabel');
const genAudio = document.getElementById('genAudio');
const playBtn = document.getElementById('play');
const stopBtn = document.getElementById('stop');
const downloadBtn = document.getElementById('download');
const audioFreq = document.getElementById('audioFreq');
const audioFreqLabel = document.getElementById('audioFreqLabel');
const audioStatus = document.getElementById('audioStatus');
const dragDrop = document.getElementById('dragDrop');
const hiddenFile = document.getElementById('hiddenFile');
const waveThumb = document.getElementById('waveThumb');
const specThumb = document.getElementById('specThumb');
const segmentsEl = document.getElementById('segments');
const matchesEl = document.getElementById('matches');
```

```
let analogueAttached = false;
let readerRunning = false;
let currentPattern = null;
let audioBufferUrl = null;
let audioCtx = null;
let sourceNode = null;

// UI helpers
resFreq.addEventListener('input', ()=>{ resLabel.textContent =
resFreq.value + ' Hz'; });
audioFreq.addEventListener('input', ()=>{ audioFreqLabel.textContent =
audioFreq.value + ' Hz'; });

function setIndicators(){
  analogueIndicator.className = 'indicator' + (analogueAttached? ' on' :
'');
  readerIndicator.className = 'indicator' + (readerRunning? ' on' : '');
  statusLabel.textContent = `Analogue: ${analogueAttached?
'Connected' : 'Disconnected'} Â·  Reader: ${readerRunning? 'Online' :
'Offline'}`;
}
setIndicators();

// -------------------- Conversion logic --------------------
function isConsonant(ch){ return /[bcdfghjklmnpqrstvwxyz]/i.test(ch); }
function vowelInsert(wort){
  const vowels = ['a','e','i','o','u'];
  let out = '';
  let vix = 0;
  for(let i=0;i< wort.length;i++){
    out += wort[i];
    if(i < wort.length - 1 && isConsonant(wort[i]) &&
isConsonant(wort[i+1])){
      out += vowels[vix % vowels.length];
      vix++;
    }
  }
  return out.replace(/x/g, ' ');
}

function convertInput(str){
  const key = (str||'').trim().toLowerCase();
  if(!key) return {text: '(empty input)', pattern: null};
  if(db[key]) return {text: db[key].text, pattern: db[key].pattern.slice()};
  const cleaned = key.replace(/[^a-z0-9]/g,'');
  const words = cleaned.match(/.{1,6}/g) || [cleaned];
  const generated = words.map(w => vowelInsert(w)).join(' ');
```

```
  const patt = [];
  for(let i=0;i< generated.length;i++) patt.push(120 +
(generated.charCodeAt(i) % 800));
  return {text: generated, pattern: patt};
}

convertBtn.addEventListener('click', ()=>{
  const res = convertInput(input.value);
  currentPattern = res.pattern;
  output.textContent = res.text + (res.pattern ? `\n\n[pattern length: $
{res.pattern.length}]` : '');
});

// -------------------- Drag & drop / file handling --------------------
['dragenter','dragover'].forEach(ev => dragDrop.addEventListener(ev, e
=> { e.preventDefault(); dragDrop.style.borderColor = '#6ee7b7'; }));
['dragleave','drop'].forEach(ev => dragDrop.addEventListener(ev, e =>
{ e.preventDefault(); dragDrop.style.borderColor = ''; }));
dragDrop.addEventListener('click', () => hiddenFile.click());
hiddenFile.addEventListener('change', () => { if(hiddenFile.files &&
hiddenFile.files[0]) handleFile(hiddenFile.files[0]); });
dragDrop.addEventListener('drop', (e) => { if(e.dataTransfer &&
e.dataTransfer.files && e.dataTransfer.files[0])
handleFile(e.dataTransfer.files[0]); });

async function handleFile(file){
  audioStatus.textContent = 'Loading file...';
  try{
    const arrayBuffer = await file.arrayBuffer();
    if(!audioCtx) audioCtx = new (window.AudioContext ||
window.webkitAudioContext)();
    const decoded = await audioCtx.decodeAudioData(arrayBuffer.slice(0));
    const channel = decoded.numberOfChannels ?
decoded.getChannelData(0) : decoded.getChannelData(0);
    drawWaveform(channel);
    drawSpectrogramApprox(channel, decoded.sampleRate);
    const energyPattern = envelopeToPattern(channel);
    currentPattern = reducePattern(energyPattern, 32);
    analogueAttached = true; setIndicators();
    const segments = segmentPattern(currentPattern);
    segmentsEl.textContent = segments.map((s,i)=>`[${i+1}] len:${s.length}
`).join(' | ');
    const matches = matchAgainstDB(segments);
    matchesEl.textContent = matches.map(m=> `${m.key || '(pseudo)'} â†’ $
{m.text || m.pseudo} (score:${Math.round(m.score||0)})`).join('\n');
    const transcription = matches.map(m => (m.text || m.pseudo || '') ).join('
');
    output.textContent = `File processed. Transcription:\n${transcription}`;
```

```
    audioStatus.textContent = 'Digital analogue processed';
    audioBufferUrl = null; // regenerate on demand
  }catch(err){
    console.error(err);
    audioStatus.textContent = 'Failed to process file';
  }
}

function envelopeToPattern(channelData){
  const win = Math.max(256, Math.floor(channelData.length / 256));
  const patt = [];
  for(let i=0;i< channelData.length;i+=win){
    let sum=0, n=0;
    for(let j=i;j< i+win && j< channelData.length;j++){ sum +=
Math.abs(channelData[j]); n++; }
    patt.push(n? (sum/n) : 0);
  }
  return patt;
}

function reducePattern(patt, targetLen=16){
  const out = [];
  const step = Math.max(1, Math.floor(patt.length / targetLen));
  for(let i=0;i< patt.length;i+=step) out.push(Math.round(120 +
Math.min(1,patt[i]*12)*1080));
  return out;
}

function segmentPattern(patt){
  if(!patt || !patt.length) return [];
  const segments = [];
  let cur = [];
  const mean = patt.reduce((a,b)=> a+b,0) / patt.length;
  const thresh = mean * 0.85;
  for(let i=0;i< patt.length;i++){
    if(patt[i] > thresh) { cur.push(patt[i]); }
    else { if(cur.length){ segments.push(cur.slice()); cur = []; } }
  }
  if(cur.length) segments.push(cur);
  if(segments.length === 0) segments.push(patt.slice(0, Math.min(8,
patt.length)));
  return segments;
}

function matchAgainstDB(segments){
  const results = [];
  const dbPatterns = Object.keys(db).map(k => ({ key: k, pattern:
db[k].pattern.map(x => Math.round(120 + (x % 800))), text: db[k].text }));
```

```
  for(const seg of segments){
    let best = { key: null, score: Infinity, text: null };
    for(const d of dbPatterns){
      const sc = normalizedDistance(seg, d.pattern);
      if(sc < best.score){ best = { key: d.key, score: sc, text: d.text, pattern:
d.pattern }; }
    }
    if(!best.key || best.score > 300){ // fallback to pseudo
      const pseudo = patternToPseudo(seg);
      best.pseudo = vowelInsert(pseudo);
      best.text = best.pseudo;
      best.key = null;
    }
    results.push(best);
  }
  return results;
}

function normalizedDistance(a,b){
  if(!a || !b || !a.length || !b.length) return Infinity;
  const L = Math.min(a.length, b.length);
  if(L === 0) return Infinity;
  let s = 0;
  for(let i=0;i< L;i++) s += Math.abs(a[i] - b[i]);
  return s / L;
}

function patternToPseudo(p){ return (p || []).map(v =>
String.fromCharCode(97 + (v % 25))).join(''); }

// -------------------- Visualization --------------------
function drawWaveform(data){
  const c = waveThumb; const ctx = c.getContext('2d');
ctx.clearRect(0,0,c.width,c.height);
  ctx.fillStyle = '#02121a'; ctx.fillRect(0,0,c.width,c.height);
  ctx.lineWidth = 1; ctx.strokeStyle = '#4ade80'; ctx.beginPath();
  const step = Math.ceil(data.length / c.width);
  for(let i=0;i< c.width;i++){
    const v = data[i*step] || 0;
    const y = (1 - (v + 1) / 2) * c.height;
    if(i===0) ctx.moveTo(i,y); else ctx.lineTo(i,y);
  }
  ctx.stroke();
}

function drawSpectrogramApprox(channel, sampleRate){
  const c = specThumb; const ctx = c.getContext('2d');
ctx.clearRect(0,0,c.width,c.height);
```

```
  const sliceLen = Math.floor(channel.length / c.width) || 1024;
  for(let x=0;x< c.width;x++){
    const start = x * sliceLen; let sum = 0; let n = 0;
    for(let i=start;i< start+sliceLen && i< channel.length;i++){ sum +=
Math.abs(channel[i]); n++; }
    const v = n ? Math.min(1, (sum/n) * 12) : 0;
    const h = Math.round(v * c.height);
    ctx.fillStyle = `rgb(${Math.round(20+v*200)},${Math.round(30+v*120)},$
{Math.round(40+v*80)})`;
    ctx.fillRect(x, c.height - h, 1, h);
  }
}

// -------------------- Audio generation --------------------
function buildWavFromPattern(pattern, baseHz){
  if(!pattern || !pattern.length) return null;
  const sampleRate = 44100;
  const seconds = Math.max(0.4, Math.min(8, pattern.length * 0.04));
  const length = Math.floor(sampleRate * seconds);
  const floatBuffer = new Float32Array(length);
  for(let i=0;i< length;i++){
    let s = 0;
    for(let j=0;j< Math.min(6, pattern.length);j++){
      const freq = baseHz * (pattern[j] / 220);
      s += Math.sin(2*Math.PI*freq*(i/sampleRate)) * (0.3 / (j+1));
    }
    floatBuffer[i] = s / 1.5;
  }
  // create WAV
  const wavBuffer = new ArrayBuffer(44 + floatBuffer.length * 2);
  const view = new DataView(wavBuffer);
  function writeString(view, offset, string){ for(let i=0;i< string.length;i++)
view.setUint8(offset + i, string.charCodeAt(i)); }
  writeString(view, 0, 'RIFF');
  view.setUint32(4, 36 + floatBuffer.length * 2, true);
  writeString(view, 8, 'WAVE');
  writeString(view, 12, 'fmt ');
  view.setUint32(16, 16, true);
  view.setUint16(20, 1, true);
  view.setUint16(22, 1, true);
  view.setUint32(24, sampleRate, true);
  view.setUint32(28, sampleRate * 2, true);
  view.setUint16(32, 2, true);
  view.setUint16(34, 16, true);
  writeString(view, 36, 'data');
  view.setUint32(40, floatBuffer.length * 2, true);
  let offset = 44;
  for(let i=0;i< floatBuffer.length;i++){
```

```
    const s = Math.max(-1, Math.min(1, floatBuffer[i]));
    view.setInt16(offset, s < 0 ? s * 0x8000 : s * 0x7FFF, true);
    offset += 2;
  }
  return new Blob([view], { type: 'audio/wav' });
}

genAudio.addEventListener('click', ()=>{
  if(!currentPattern || !currentPattern.length){ audioStatus.textContent =
'No pattern available â€” attach digital analogue or convert text first';
return; }
  const baseHz = parseInt(audioFreq.value, 10) || 440;
  const wav = buildWavFromPattern(currentPattern, baseHz);
  if(!wav){ audioStatus.textContent = 'Audio generation failed'; return; }
  if(audioBufferUrl) URL.revokeObjectURL(audioBufferUrl);
  audioBufferUrl = URL.createObjectURL(wav);
  audioStatus.textContent = 'Audio generated';
});

playBtn.addEventListener('click', async ()=>{
  if(!audioBufferUrl){ audioStatus.textContent = 'No audio â€” generate
first'; return; }
  if(!audioCtx) audioCtx = new (window.AudioContext ||
window.webkitAudioContext)();
  if(sourceNode) try{ sourceNode.stop(); }catch(e){}
  const resp = await fetch(audioBufferUrl);
  const ab = await resp.arrayBuffer();
  const decoded = await audioCtx.decodeAudioData(ab.slice(0));
  sourceNode = audioCtx.createBufferSource();
  sourceNode.buffer = decoded;
sourceNode.connect(audioCtx.destination); sourceNode.start();
  audioStatus.textContent = 'Playing';
  sourceNode.onended = ()=> audioStatus.textContent = 'Playback
finished';
});

stopBtn.addEventListener('click', ()=>{ if(sourceNode)
try{ sourceNode.stop(); audioStatus.textContent = 'Stopped'; }catch(e){} });

downloadBtn.addEventListener('click', ()=>{ if(!audioBufferUrl)
{ audioStatus.textContent = 'No audio to download'; return; } const a =
document.createElement('a'); a.href = audioBufferUrl; a.download =
'aanic_audio.wav'; document.body.appendChild(a); a.click();
a.remove(); });

// -------------------- Keyboard shortcuts & attach behavior
--------------------
attachAnalogue.addEventListener('click', ()=>{ hiddenFile.click(); });
```

```
window.addEventListener('keydown', e => { if(e.ctrlKey && e.shiftKey &&
e.key.toLowerCase() === 'a') hiddenFile.click(); });

// -------------------- Small helper used in matching fallback
--------------------
function patternToPseudo(p){ return (p || []).map(v =>
String.fromCharCode(97 + (v % 25))).join(''); }

</script>


 s



  <meta name="viewport" content="width=device-width,initial-scale=1">
  <title>AugVT â€” Free Mobile Phone (Interactive Demo)</title>
  <style>
    :root{--bg:#071021;--card:#0e1724;--accent:#00e6a8;--
muted:#9fb0bf}
    body{margin:0;font-family:Inter,system-ui,Segoe
UI,Arial;background:linear-gradient(180deg,#041026 0%,#06152a
100%);color:#eaf6f0}
    .app{display:grid;grid-template-columns:300px
1fr;gap:12px;height:100vh;padding:12px}
    .panel{background:linear-
gradient(180deg,rgba(255,255,255,0.02),transparent);border-
radius:12px;padding:12px}
    header{grid-column:1/-1;display:flex;align-
items:center;gap:12px;padding:12px}
    h1{margin:0;font-size:18px}
    .contacts{display:flex;flex-direction:column;gap:8px;height:calc(100vh
- 140px);overflow:auto}
    .contact{padding:10px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.02);cursor:pointer}
    .contact.active{background:rgba(0,0,0,0.25)}
    .chat{display:flex;flex-direction:column;height:calc(100vh - 140px)}
    .messages{flex:1;overflow:auto;padding:12px;display:flex;flex-
direction:column;gap:8px}
    .msg{max-width:70%;padding:8px;border-radius:8px}
    .out{align-self:flex-end;background:linear-
gradient(90deg,#005f4d,#008c6a)}
    .in{align-self:flex-start;background:rgba(255,255,255,0.04)}
    .composer{display:flex;gap:8px;padding:8px}
    input[type=text]{flex:1;padding:10px;border-radius:8px;border:1px
solid rgba(255,255,255,0.03);background:transparent;color:inherit}
    button{padding:10px 12px;border-
```

```
radius:8px;border:0;background:var(--accent);color:#012;cursor:pointer}
  .small{font-size:12px;color:var(--muted)}
  .controls{display:flex;gap:8px;align-items:center}

textarea{width:100%;height:80px;background:transparent;color:inherit;b
order:1px solid rgba(255,255,255,0.03);border-radius:8px;padding:8px}
  .cols{display:grid;grid-template-columns:1fr 1fr;gap:8px}
  @media(max-width:900px){.app{grid-template-
columns:1fr} .contacts{height:200px}.chat{height:calc(100vh - 330px)}}
 </style>


 <header class="panel">
  <h1>AugVT â€” Free Mobile Phone (Demo)</h1>
  <div style="margin-left:auto" class="small">No servers required â€”
manual WebRTC signalling (copy/paste) or local simulated network.
Stored locally.</div>
 </header>
 <div class="app">
  <aside class="panel">
   <div style="display:flex;gap:8px;margin-bottom:8px">
    <input id="contactName" type="text" placeholder="New contact
name">
    <button id="addContact">Add</button>
   </div>
   <div class="contacts panel" id="contacts"></div>

   <div style="margin-top:8px" class="small">Shareable Contact Token
(copy to share):</div>
   <div style="display:flex;gap:8px;margin-top:6px">
    <input id="myToken" type="text" readonly="">
    <button id="regenToken">Regenerate</button>
   </div>

   <div style="margin-top:10px">
    <div class="small">Quick Connect (WebRTC manual signalling)</
div>
    <div class="small" style="margin-top:6px">1) Create Offer - &gt;
Copy - &gt; Send to peer. 2) Paste their Answer - &gt; Connect.</div>
    <div style="margin-top:6px" class="cols">
     <div>
      <button id="createOffer">Create Offer</button>
      <textarea id="offer" placeholder="Offer will appear here"></
textarea>
     </div>
     <div>
      <button id="createAnswer">Create Answer</button>
      <textarea id="answer" placeholder="Paste offer here then click
```