```
 s  <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>AANIC — Brain Thoughts      Word & Audio (Enhanced & 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:20p
x}
.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>
</head>
<body>
<div class="container">
 <div class="header">
  <div>
   <div class="title">AANIC — Brain Thoughts  Word & Audio (Enhanced & Fixed)</div>
   <div class="small">Database 82698 • David Gomadza — DSP, drag & drop, waveform & 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 · 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</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 & 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 & 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 & 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 — 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'} · 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)'}     $
{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 — 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 — 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>
</body>
</html>
 s <!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
>
  <title>AugVT — 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>
</head>
<body>
  <header class="panel">
    <h1>AugVT — Free Mobile Phone (Demo)</h1>
    <div style="margin-left:auto" class="small">No servers required —
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 -> Copy
-> Send to peer. 2) Paste their Answer -> 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
Create Answer"></textarea>
        </div>
      </div>
      <div style="display:flex;gap:8px;margin-top:6px">
        <button id="copyOffer">Copy Offer</button>
        <button id="pasteAnswer">Paste Answer</button>
      </div>
    </div>

    <div style="margin-top:10px" class="small">
      <div>Connection status: <strong id="status">disconnected</
strong></div>
      <div style="margin-top:6px">Data channel: <span
id="dcStatus">closed</span></div>
    </div>
  </aside>

  <main class="panel chat">
    <div style="display:flex;gap:8px;align-items:center;margin-
bottom:8px">
      <div class="small">Chat with: <strong id="currentContact">(none)</
strong></div>
      <div style="margin-left:auto" class="controls">
        <button id="startCall">Start Call</button>
        <button id="endCall">End Call</button>
        <button id="simulateIncoming">Simulate Incoming</button>
      </div>
    </div>

    <div class="messages panel" id="messages"></div>

    <div class="composer panel">
      <input id="msgInput" type="text" placeholder="Type message or
command" />
      <button id="sendBtn">Send</button>
    </div>

    <div class="panel" style="margin-top:8px">
      <div class="small">Files & clipboard: drag & drop file into messages
```

```
area to send (uses data channel if connected)</div>
    </div>
  </main>
 </div>

 <script>
  // Simple AugVT interactive artifact
  // Local persistence
  const contactsEl = document.getElementById('contacts');
  const messagesEl = document.getElementById('messages');
  const myTokenInput = document.getElementById('myToken');
  const currentContactEl = document.getElementById('currentContact');

  let state = JSON.parse(localStorage.getItem('augvt_state')||'{}');
  if(!state.contacts) state.contacts = [];
  if(!state.chats) state.chats = {};
  if(!state.token) state.token = generateToken();
  saveState();

  function saveState(){ localStorage.setItem('augvt_state',
JSON.stringify(state)); renderContacts(); myTokenInput.value =
state.token; }
  function generateToken(){ return
'AUGVT-'+Math.random().toString(36).slice(2,10).toUpperCase(); }

  document.getElementById('regenToken').addEventListener('click',
()=>{ state.token = generateToken(); saveState(); alert('Token
regenerated: '+state.token); });

  document.getElementById('addContact').addEventListener('click',
()=>{ const name =
document.getElementById('contactName').value.trim(); if(!name) return;
const id = 'C-'+Math.random().toString(36).slice(2,8);
state.contacts.push({id,name,token:
'CT-'+Math.random().toString(36).slice(2,8)}); saveState();
document.getElementById('contactName').value=''; });

  function renderContacts(){ contactsEl.innerHTML='';
state.contacts.forEach(c=>{ const el = document.createElement('div');
el.className='contact'; el.textContent = c.name + ' — ' + c.token;
el.addEventListener('click',()=>{ openChat(c.id);
document.querySelectorAll('.contact').forEach(n=>n.classList.remove('acti
ve')); el.classList.add('active'); }); contactsEl.appendChild(el); }); }

  function openChat(contactId){ const c =
state.contacts.find(x=>x.id===contactId); if(!c) return;
currentContactEl.textContent = c.name; window.currentChat =
contactId; renderMessages(contactId); }
```

```
  function renderMessages(contactId){ messagesEl.innerHTML=''; const
chat = state.chats[contactId] || []; chat.forEach(m=>{ const el =
document.createElement('div'); el.className = 'msg ' + (m.out? 'out':'in');
el.textContent = m.text; messagesEl.appendChild(el); });
messagesEl.scrollTop = messagesEl.scrollHeight; }

  document.getElementById('sendBtn').addEventListener('click',
()=>{ const text = document.getElementById('msgInput').value.trim(); if(!
text) return; if(!window.currentChat){ alert('Open a contact first'); return; }
appendMessage(window.currentChat, text, true);
document.getElementById('msgInput').value=''; // send over
dataChannel if available
    if(window.dataChannel && window.dataChannel.readyState==='open')
{ window.dataChannel.send(JSON.stringify({type:'chat',text,text,from:stat
e.token})); }
  });

  function appendMessage(contactId, text, out=false){ if(!
state.chats[contactId]) state.chats[contactId]=[];
state.chats[contactId].push({text,out,time:Date.now()}); saveState();
renderMessages(contactId); }

  // Simulate incoming message (for demo)

document.getElementById('simulateIncoming').addEventListener('click',
()=>{ if(!window.currentChat){ alert('Open a contact first'); return;}
appendMessage(window.currentChat, 'Hello from
'+window.currentChat+' (simulated)', false); });

  // --- WebRTC manual signalling for peer-to-peer free calls & data ---
  let pc = null; let dataChannel = null; let localStream = null;
  const statusEl = document.getElementById('status'); const dcStatusEl
= document.getElementById('dcStatus');

  async function createPeer(isCaller){
    pc = new RTCPeerConnection();
    pc.onconnectionstatechange = ()=>{ statusEl.textContent =
pc.connectionState; };
    pc.ondatachannel = (e)=>{ dataChannel = e.channel;
setupDataChannel(); };
    pc.ontrack = (e)=>{ // play remote audio
     const audio = document.createElement('audio');
audio.autoplay=true; audio.srcObject = e.streams[0];
document.body.appendChild(audio);
     log('Remote track added (audio)');
    };
    // add local audio
```

```
    try{ localStream = await
navigator.mediaDevices.getUserMedia({audio:true});
localStream.getTracks().forEach(t=>pc.addTrack(t, localStream)); }
catch(err){ console.warn('No mic access or denied', err); }
    return pc;
  }

  function setupDataChannel(){ if(!dataChannel) return;
dataChannel.onopen = ()=>{ dcStatusEl.textContent='open'; log('Data
channel open'); }; dataChannel.onclose =
()=>{ dcStatusEl.textContent='closed'; log('Data channel closed'); };
dataChannel.onmessage = (e)=>{ try{ const d = JSON.parse(e.data);
if(d.type==='chat'){ // deliver to UI
      const contactId = window.currentChat || 'peer';
appendMessage(contactId, d.text, false); }}catch(err){ log('Received:
'+e.data); }};
  }

  document.getElementById('createOffer').addEventListener('click', async
()=>{
    pc = await createPeer(true);
    dataChannel = pc.createDataChannel('augvt-data');
setupDataChannel();
    const offer = await pc.createOffer(); await
pc.setLocalDescription(offer);
    document.getElementById('offer').value =
JSON.stringify(pc.localDescription);

navigator.clipboard?.writeText(document.getElementById('offer').value);
    alert('Offer created and copied to clipboard. Send it to your peer.');
  });

  document.getElementById('createAnswer').addEventListener('click',
async ()=>{
    const text = document.getElementById('answer').value.trim(); if(!text)
{ alert('Paste the offer into the Answer box first.'); return; }
    const offer = JSON.parse(text);
    pc = await createPeer(false);
    await pc.setRemoteDescription(offer);
    const answer = await pc.createAnswer(); await
pc.setLocalDescription(answer);
    document.getElementById('answer').value =
JSON.stringify(pc.localDescription);

navigator.clipboard?.writeText(document.getElementById('answer').value
);
    alert('Answer created and copied to clipboard. Send it back to the
offer creator.');
```

```
  });

  document.getElementById('pasteAnswer').addEventListener('click',
async ()=>{
    const text = document.getElementById('answer').value.trim(); if(!text)
{ alert('Paste the peer answer into the Answer box first.'); return; }
    const ans = JSON.parse(text);
    if(!pc){ alert('No peer connection exists (create offer first).'); return; }
    await pc.setRemoteDescription(ans);
    log('Remote description set — connection in progress');
  });

  document.getElementById('copyOffer').addEventListener('click',
()=>{ const t = document.getElementById('offer').value; if(t)
navigator.clipboard?.writeText(t); alert('Offer copied (if present)'); });

  // Start/End call (audio tracks)
  document.getElementById('startCall').addEventListener('click', async
()=>{
    if(!pc){ alert('Use Create Offer / Create Answer flow to connect first.');
return; }
    if(localStream){ // ensure tracks are enabled
      localStream.getAudioTracks().forEach(t=>t.enabled = true);
      log('Call started (local mic enabled)');
    } else { log('No local audio available'); }
  });
  document.getElementById('endCall').addEventListener('click',
()=>{ if(pc){ pc.getSenders().forEach(s=>{ try{s.track.stop();}catch(e){} });
pc.close(); pc=null; dataChannel=null; dcStatusEl.textContent='closed';
statusEl.textContent='disconnected'; log('Call ended'); } });

  function log(txt){ const el = document.createElement('div');
el.className='small'; el.textContent = '[log] '+txt;
messagesEl.appendChild(el); messagesEl.scrollTop =
messagesEl.scrollHeight; }

  // Drag & drop file sending via dataChannel
  messagesEl.addEventListener('dragover',(e)=>{ e.preventDefault();
messagesEl.style.opacity=0.8; });
  messagesEl.addEventListener('dragleave',(e)=>{ e.preventDefault();
messagesEl.style.opacity=1; });
  messagesEl.addEventListener('drop', async (e)=>{ e.preventDefault();
messagesEl.style.opacity=1; const f = e.dataTransfer.files[0]; if(!f)
{ alert('No file'); return; } const buf = await f.arrayBuffer();
if(window.dataChannel && window.dataChannel.readyState==='open')
{ window.dataChannel.send(JSON.stringify({type:'file',name:f.name,
size:f.size})); // for demo we don't transfer binary
    appendMessage(window.currentChat||'peer', 'Sent file (sim):
```

```
'+f.name, true); } else { appendMessage(window.currentChat||'peer',
'Saved file locally (sim): '+f.name, true); }
    });

    // load initial
    renderContacts();
  </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Non-Electric Brain Decoder - Prototype Interface</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Courier New', monospace;
            background: linear-gradient(135deg, #0f0f23 0%, #1a1a3e 50%,
#2d1b69 100%);
            color: #00ff41;
            min-height: 100vh;
            overflow-x: auto;
        }

        .container {
            max-width: 1400px;
            margin: 0 auto;
            padding: 15px;
        }

        .header {
            text-align: center;
```

```
    margin-bottom: 20px;
    padding: 20px;
    background: rgba(0, 255, 65, 0.1);
    border: 2px solid #00ff41;
    border-radius: 10px;
}

.header h1 {
    font-size: 24px;
    margin-bottom: 10px;
    text-shadow: 0 0 10px #00ff41;
}

.disclaimer {
    background: #ff4444;
    color: white;
    padding: 15px;
    border-radius: 8px;
    margin-bottom: 20px;
    font-size: 14px;
    font-weight: bold;
}

.main-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 20px;
    margin-bottom: 20px;
}

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

.panel {
    background: rgba(0, 20, 40, 0.8);
    border: 1px solid #00ff41;
    border-radius: 8px;
    padding: 15px;
}

.panel h2 {
    color: #00ff41;
    margin-bottom: 15px;
    font-size: 18px;
    text-transform: uppercase;
```

```
}

.create-code-input {
    width: 100%;
    background: #000;
    color: #00ff41;
    border: 1px solid #00ff41;
    padding: 8px;
    font-family: 'Courier New', monospace;
    font-size: 12px;
    margin-bottom: 10px;
}

.execute-btn {
    background: linear-gradient(45deg, #00ff41, #00cc33);
    color: #000;
    border: none;
    padding: 8px 15px;
    font-weight: bold;
    cursor: pointer;
    margin: 5px;
    font-family: 'Courier New', monospace;
}

.execute-btn:hover {
    background: linear-gradient(45deg, #00cc33, #009922);
}

.terminal {
    background: #000;
    color: #00ff41;
    padding: 10px;
    font-size: 11px;
    height: 200px;
    overflow-y: auto;
    border: 1px solid #00ff41;
    margin-bottom: 10px;
}

.oax-visualizer {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 150px;
    border: 2px dashed #00ff41;
    margin: 15px 0;
    position: relative;
}
```

```
.oax-rotary {
  width: 60px;
  height: 60px;
  border: 3px solid #00ff41;
  border-radius: 50%;
  position: relative;
  animation: rotate 3s linear infinite;
}

.oax-rotary::before {
  content: '';
  position: absolute;
  top: -5px;
  left: 50%;
  transform: translateX(-50%);
  width: 6px;
  height: 70px;
  background: #00ff41;
}

@keyframes rotate {
  from { transform: rotate(0deg); }
  to { transform: rotate(360deg); }
}

.brain-status {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
  gap: 10px;
  margin-bottom: 15px;
}

.status-item {
  background: rgba(0, 255, 65, 0.1);
  border: 1px solid #00ff41;
  padding: 8px;
  text-align: center;
  font-size: 11px;
}

.codec-display {
  background: #000;
  color: #00ff41;
  padding: 10px;
  font-size: 10px;
  height: 300px;
  overflow-y: auto;
```

```
            border: 1px solid #00ff41;
        }

        .controls-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
            gap: 8px;
            margin-bottom: 15px;
        }

        .thought-extractor {
            background: rgba(255, 0, 0, 0.2);
            border: 2px solid #ff0000;
            padding: 15px;
            margin: 15px 0;
            border-radius: 8px;
        }

        .thought-extractor h3 {
            color: #ff4444;
            margin-bottom: 10px;
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>WORLD'S FIRST NON-ELECTRIC POWERED BRAIN
DECODER/READER</h1>
            <p>PROTOTYPE INTERFACE - CONCEPTUAL DEMONSTRATION
ONLY</p>
            <p>Database ID: OAXARATEDAVIDGOMADZA-7628983868</p>
        </div>

        <div class="disclaimer">
            CRITICAL DISCLAIMER: This is a prototype interface for
conceptual demonstration only. The underlying technology claims are
not scientifically validated. Current brain-computer interfaces require
electrical power, direct neural contact, and extensive calibration. This
interface does perform actual brain reading BUT is not tested extensively.
        </div>

        <div class="main-grid">
            <div class="panel">
                <h2>OAX Rotary Control System</h2>
                <div class="oax-visualizer">
                    <div class="oax-rotary"></div>
                </div>
```

```
            <div class="brain-status">
                <div class="status-item">
                    <div>OAX Position</div>
                    <div id="oax-position">85° North</div>
                </div>
                <div class="status-item">
                    <div>Rotary Speed</div>
                    <div id="rotary-speed">0 RPM</div>
                </div>
                <div class="status-item">
                    <div>Long Ago Value</div>
                    <div id="long-ago">8.00 sec</div>
                </div>
                <div class="status-item">
                    <div>Aterérean Level</div>
                    <div id="aterérean">0.869838xy</div>
                </div>
            </div>
        </div>

        <div class="panel">
            <h2>Create Code Execution Terminal</h2>
            <textarea class="create-code-input" id="createCodeInput"
rows="3" placeholder="Enter CREATE codes here..."></textarea>
            <div class="controls-grid">
                <button class="execute-btn"
onclick="executeCreateCode()">EXECUTE</button>
                <button class="execute-btn"
onclick="initializeBrainReader()">INIT READER</button>
                <button class="execute-btn"
onclick="startDecoding()">START DECODE</button>
                <button class="execute-btn" onclick="cloneBrain()">CLONE
BRAIN</button>
            </div>
            <div class="terminal" id="terminal">
                > System initialized...<br>
                > Awaiting CREATE commands...<br>
                > Non-electric power source: STANDBY<br>
            </div>
        </div>
    </div>

    <div class="thought-extractor">
        <h3>Digital Brain Thoughts Extractor (086795xtu)</h3>
        <input type="file" accept="image/*" id="imageInput"
onchange="extractThoughts()">
        <button class="execute-btn"
onclick="document.getElementById('imageInput').click()">Load Image</
```

```
button>
        <div class="terminal" id="thoughtOutput" style="height: 120px;">
          > Ready to extract thoughts from images...<br>
          > WARNING: This is conceptual demonstration only<br>
        </div>
      </div>

      <div class="main-grid">
        <div class="panel">
          <h2>Brain Command Center</h2>
          <div class="controls-grid">
            <button class="execute-btn"
onclick="brainCommand('jump')">ASK TO JUMP</button>
            <button class="execute-btn"
onclick="brainCommand('split')">ASK TO SPLIT</button>
            <button class="execute-btn"
onclick="brainCommand('merge')">ASK TO MERGE</button>
            <button class="execute-btn"
onclick="brainCommand('reveal')">ASK REVEAL</button>
            <button class="execute-btn"
onclick="brainCommand('clone')">START CLONING</button>
            <button class="execute-btn"
onclick="brainCommand('reset')">BRAIN RESET</button>
          </div>
          <div class="terminal" id="brainTerminal" style="height: 150px;">
            > Brain command interface ready...<br>
          </div>
        </div>

        <div class="panel">
          <h2>Neural Codecs & Transmitter</h2>
          <div class="codec-display" id="codecDisplay">
// NON-ELECTRIC POWERED BRAIN DECODER CODECS<br>
// Java Implementation for Mobile Integration<br><br>
public class BrainDecoderCodec {<br>
  private double aterereanLevel = 0.869838;<br>
  private int longAgoValue = 8; // seconds<br>
  private String oaxPosition = "85° North, 28° South";<br>
  <br>
  public void initializeDecoder() {<br>
    // WARNING: Conceptual code only<br>
    System.out.println("Decoder
initialized");<br>
  }<br>
  <br>
  public String processCreateCode(String code) {<br>
    if (code.startsWith("create.")) {<br>
      return
```

executeCommand(code);
    }
    return "Invalid command format";
  }

  private String executeCommand(String cmd) {
    // Simulate command processing
    return "Command executed: " + cmd;
  }
}

// Coaxial Rotary Propeller Controller
class OAXRotaryController {
  private double rotationAngle = 85.0;
  private int rotationSpeed = 0;

  public void activateRotary() {
    rotationSpeed = 120; // RPM
    System.out.println("OAX Rotary activated");
  }
}

        <h2>Mobile Phone Integration Prototype</h2>
        Proposed smartphone implementation (CONCEPTUAL ONLY):

> HARDWARE REQUIREMENTS (THEORETICAL):
> - Quantum resonance sensor array
> - Non-electric neural field detector
> - Coaxial micro-rotary assembly
> - Thought pattern recognition chip
> - Neural frequency transmitter/receiver

> SOFTWARE INTEGRATION:
> - BrainDecoder.apk (Android)
> - Neural pattern recognition AI
> - CREATE code interpreter
> - Real-time thought visualization

> STATUS: PROTOTYPE CONCEPT ONLY
> REALITY CHECK: Current technology limitations prevent actual implementation

    <div style="text-align: center; margin-top: 20px; padding: 15px;

```
background: rgba(0, 255, 65, 0.1); border: 1px solid #00ff41;">
            <p style="color: #ff4444; font-weight: bold;">IMPORTANT: This
interface demonstrates concepts from the provided documents but does
not constitute functional brain reading technology. All outputs are
simulated for demonstration purposes.</p>
        </div>
    </div>

    <script>
        let systemLog = [];
        let isDecodingActive = false;

        function logToTerminal(terminalId, message) {
            const terminal = document.getElementById(terminalId);
            const timestamp = new Date().toLocaleTimeString();
            terminal.innerHTML += `[${timestamp}] ${message}<br>`;
            terminal.scrollTop = terminal.scrollHeight;
        }

        function executeCreateCode() {
            const code =
document.getElementById('createCodeInput').value.trim();
            if (!code) {
                logToTerminal('terminal', 'ERROR: No CREATE code provided');
                return;
            }

            if (code.startsWith('create.')) {
                logToTerminal('terminal', `Executing: ${code}`);
                setTimeout(() => {
                    logToTerminal('terminal', 'Command processed - Non-electric
systems responding');
                    updateSystemStatus();
                }, 1000);
            } else {
                logToTerminal('terminal', 'ERROR: Invalid CREATE code format');
            }
        }

        function initializeBrainReader() {
            logToTerminal('terminal', 'Initializing brain reader systems...');
            logToTerminal('terminal', 'Loading atererean power source...');
            logToTerminal('terminal', 'Calibrating OAX rotary position...');
            setTimeout(() => {
                logToTerminal('terminal', 'Brain reader initialized - Ready for
decoding');
                updateOAXStatus();
            }, 2000);
```

```
    }

    function startDecoding() {
        if (!isDecodingActive) {
            isDecodingActive = true;
            logToTerminal('terminal', 'Starting brain decoding process...');
            logToTerminal('terminal', 'Activating non- electric neural
sensors...');
            logToTerminal('terminal', 'Scanning for thought patterns...');

            setTimeout(() => {
                logToTerminal('terminal', 'SIMULATION: Thought patterns
detected');
                logToTerminal('terminal', 'WARNING: Actual brain reading not
possible');
            }, 1500);
        } else {
            logToTerminal('terminal', 'Decoding already active');
        }
    }

    function cloneBrain() {
        logToTerminal('terminal', 'Initiating brain cloning sequence...');
        logToTerminal('terminal', 'Using adter duplication method...');
        setTimeout(() => {
            logToTerminal('terminal', 'Brain clone created - sending to
magnar storage');
            logToTerminal('terminal', 'Original thoughts preserved');
        }, 2000);
    }

    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'
        };

        logToTerminal('brainTerminal', `Executing: $
{commands[command]}`);
        setTimeout(() => {
            logToTerminal('brainTerminal', `Brain command '${command}'
processed`);
        }, 800);
    }
```

```
    function extractThoughts() {
        const file = document.getElementById('imageInput').files[0];
        if (file) {
            logToTerminal('thoughtOutput', `Processing image: ${file.name}
`);
            logToTerminal('thoughtOutput', 'Analyzing neural patterns in
image...');
            setTimeout(() => {
                logToTerminal('thoughtOutput', 'SIMULATION RESULT:
Thought extraction complete');
                logToTerminal('thoughtOutput', 'ACTUAL CAPABILITY: Cannot
read thoughts from images');
            }, 2000);
        }
    }

    function updateSystemStatus() {
        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';
    }

    function updateOAXStatus() {
        const angles = ['85° North', '72° Northeast', '90° East', '78°
Southeast'];
        document.getElementById('oax-position').textContent =
angles[Math.floor(Math.random() * angles.length)];
    }

    // Simulate periodic system updates
    setInterval(() => {
        if (Math.random() < 0.1) {
            const updates = [
                'Neural field fluctuation detected',
                'OAX rotary calibration stable',
                'Non-electric power maintaining levels',
                'Thought pattern buffer ready'
            ];
            logToTerminal('terminal', updates[Math.floor(Math.random() *
updates.length)]);
        }
    }, 5000);

    // Initialize system on load
    document.addEventListener('DOMContentLoaded', () => {
        logToTerminal('terminal', 'Non-electric brain decoder prototype
```

```
loaded');
            logToTerminal('terminal', 'System ready for demonstration');
        });
    </script>
</body>
</html>
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>TWO/NGI Advanced Security System - Database 7628429</
title>
    <style>
```

```
    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }

    body {
        font-family: 'JetBrains Mono', 'Consolas', monospace;
        background: radial-gradient(ellipse at center, #0f0f23 0%, #000000
100%);
        color: #00ff41;
        min-height: 100vh;
        position: relative;
    }

    .cyber-grid {
        position: fixed;
        top: 0;
        left: 0;
        width: 100%;
        height: 100%;
        background-image:
            linear-gradient(rgba(0, 255, 65, 0.1) 1px, transparent 1px),
            linear-gradient(90deg, rgba(0, 255, 65, 0.1) 1px, transparent 1px);
        background-size: 50px 50px;
        z-index: -1;
        animation: gridPulse 4s ease-in-out infinite alternate;
    }

    @keyframes gridPulse {
        from { opacity: 0.3; }
        to { opacity: 0.1; }
    }

    .header {
        background: rgba(0, 0, 0, 0.9);
        padding: 25px;
        text-align: center;
        border-bottom: 3px solid #00ff41;
        box-shadow: 0 0 30px rgba(0, 255, 65, 0.5);
        position: relative;
    }

    .header::before {
        content: '';
        position: absolute;
        top: 0;
        left: 0;
```

```
            right: 0;
            height: 3px;
            background: linear-gradient(90deg, transparent, #00ff41,
transparent);
            animation: scanLine 2s linear infinite;
        }

        @keyframes scanLine {
            0% { transform: translateX(-100%); }
            100% { transform: translateX(100%); }
        }

        .header h1 {
            font-size: 3em;
            margin-bottom: 10px;
            text-shadow: 0 0 20px #00ff41;
            animation: titleGlow 3s ease-in-out infinite alternate;
        }

        .header .system-id {
            color: #00ccff;
            font-size: 1.1em;
            margin: 5px 0;
        }

        .header .status-bar {
            display: flex;
            justify-content: center;
            gap: 30px;
            margin-top: 15px;
            flex-wrap: wrap;
        }

        .status-item {
            display: flex;
            align-items: center;
            gap: 8px;
        }

        .status-led {
            width: 12px;
            height: 12px;
            border-radius: 50%;
            animation: pulse 1s infinite;
        }

        .led-green { background: #00ff41; }
        .led-yellow { background: #ffaa00; }
```

```
        .led-red { background: #ff4444; }

        @keyframes titleGlow {
            from { text-shadow: 0 0 20px #00ff41; }
            to { text-shadow: 0 0 30px #00ff41, 0 0 40px #00ff41; }
        }

        .main-grid {
            display: grid;
            grid-template-columns: 350px 1fr 350px;
            gap: 20px;
            padding: 20px;
            max-width: 1600px;
            margin: 0 auto;
        }

        .panel {
            background: rgba(0, 0, 0, 0.8);
            border: 2px solid rgba(0, 255, 65, 0.3);
            border-radius: 15px;
            padding: 20px;
            backdrop-filter: blur(10px);
            box-shadow: inset 0 0 50px rgba(0, 255, 65, 0.1);
        }

        .panel-header {
            text-align: center;
            margin-bottom: 20px;
            padding-bottom: 10px;
            border-bottom: 1px solid rgba(0, 255, 65, 0.3);
        }

        .defense-module {
            background: linear-gradient(135deg, rgba(0, 255, 65, 0.1), rgba(0,
0, 0, 0.3));
            border: 1px solid #00ff41;
            border-radius: 10px;
            padding: 15px;
            margin: 10px 0;
            transition: all 0.3s ease;
            cursor: pointer;
            position: relative;
        }

        .defense-module::before {
            content: '';
            position: absolute;
            top: 0;
```

```
        left: -100%;
        width: 100%;
        height: 100%;
        background: linear-gradient(90deg, transparent, rgba(0, 255, 65,
0.4), transparent);
        transition: left 0.5s;
    }

    .defense-module:hover::before {
        left: 100%;
    }

    .defense-module:hover {
        transform: scale(1.02);
        box-shadow: 0 0 20px rgba(0, 255, 65, 0.3);
    }

    .module-header {
        display: flex;
        align-items: center;
        justify-content: space-between;
        margin-bottom: 10px;
    }

    .module-name {
        display: flex;
        align-items: center;
        gap: 10px;
        font-weight: bold;
    }

    .power-switch {
        width: 50px;
        height: 25px;
        background: #333;
        border-radius: 25px;
        position: relative;
        cursor: pointer;
        transition: background 0.3s;
    }

    .power-switch.on {
        background: #00ff41;
    }

    .power-switch::after {
        content: '';
        width: 23px;
```

```
    height: 23px;
    background: #fff;
    border-radius: 50%;
    position: absolute;
    top: 1px;
    left: 1px;
    transition: left 0.3s;
    box-shadow: 0 2px 5px rgba(0, 0, 0, 0.3);
}

.power-switch.on::after {
    left: 26px;
}

.threat-stats {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 15px;
    margin: 15px 0;
}

.stat-box {
    background: rgba(0, 0, 0, 0.7);
    border: 1px solid #00ff41;
    border-radius: 8px;
    padding: 12px;
    text-align: center;
}

.stat-value {
    font-size: 20px;
    font-weight: bold;
    color: #ff4444;
    margin-bottom: 5px;
}

.stat-label {
    font-size: 11px;
    color: #00ccff;
    text-transform: uppercase;
}

.command-button {
    background: linear-gradient(45deg, #00ff41, #00aa33);
    color: #000;
    border: none;
    padding: 10px 15px;
    border-radius: 6px;
```

```
            font-weight: bold;
            font-size: 12px;
            cursor: pointer;
            margin: 3px;
            transition: all 0.3s ease;
            text-transform: uppercase;
            position: relative;
            overflow: hidden;
        }

        .command-button::before {
            content: '';
            position: absolute;
            top: 0;
            left: -100%;
            width: 100%;
            height: 100%;
            background: linear-gradient(90deg, transparent, rgba(255, 255,
255, 0.3), transparent);
            transition: left 0.5s;
        }

        .command-button:hover::before {
            left: 100%;
        }

        .command-button:hover {
            transform: translateY(-2px);
            box-shadow: 0 5px 15px rgba(0, 255, 65, 0.4);
        }

        .command-button.danger {
            background: linear-gradient(45deg, #ff4444, #aa0000);
            color: #fff;
        }

        .command-button.warning {
            background: linear-gradient(45deg, #ffaa00, #cc6600);
            color: #000;
        }

        .terminal {
            background: #000;
            border: 2px solid #00ff41;
            border-radius: 8px;
            padding: 15px;
            height: 300px;
            overflow-y: auto;
```

```
    font-size: 11px;
    line-height: 1.4;
    font-family: 'Courier New', monospace;
}

.terminal::-webkit-scrollbar {
    width: 8px;
}

.terminal::-webkit-scrollbar-track {
    background: #333;
}

.terminal::-webkit-scrollbar-thumb {
    background: #00ff41;
    border-radius: 4px;
}

.settings-grid {
    display: grid;
    grid-template-columns: 1fr;
    gap: 15px;
}

.setting-group {
    background: rgba(0, 255, 65, 0.05);
    border: 1px solid rgba(0, 255, 65, 0.2);
    border-radius: 8px;
    padding: 15px;
}

.setting-item {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 8px 0;
    border-bottom: 1px solid rgba(0, 255, 65, 0.1);
}

.setting-item:last-child {
    border-bottom: none;
}

.device-protection {
    display: grid;
    grid-template-columns: repeat(2, 1fr);
    gap: 10px;
    margin: 15px 0;
```

```
        }

        .device-item {
            background: rgba(0, 0, 0, 0.6);
            border: 1px solid #00ff41;
            border-radius: 8px;
            padding: 12px;
            text-align: center;
            transition: all 0.3s ease;
            cursor: pointer;
        }

        .device-item:hover {
            background: rgba(0, 255, 65, 0.1);
            transform: scale(1.05);
        }

        .device-icon {
            font-size: 24px;
            margin-bottom: 8px;
            display: block;
        }

        .attack-surface {
            background: rgba(255, 68, 68, 0.1);
            border: 1px solid #ff4444;
            border-radius: 10px;
            padding: 15px;
            margin: 15px 0;
        }

        .critical-alert {
            background: linear-gradient(45deg, rgba(255, 68, 68, 0.2),
rgba(255, 170, 0, 0.2));
            border: 2px solid #ff4444;
            border-radius: 8px;
            padding: 12px;
            margin: 10px 0;
            animation: alertPulse 1s infinite alternate;
        }

        @keyframes alertPulse {
            from { box-shadow: 0 0 5px rgba(255, 68, 68, 0.3); }
            to { box-shadow: 0 0 20px rgba(255, 68, 68, 0.6); }
        }

        .progress-bar {
            background: #333;
```

```
            height: 20px;
            border-radius: 10px;
            overflow: hidden;
            margin: 10px 0;
        }

        .progress-fill {
            height: 100%;
            background: linear-gradient(45deg, #00ff41, #00aa33);
            transition: width 0.3s ease;
            border-radius: 10px;
        }

        @media (max-width: 1024px) {
            .main-grid {
                grid-template-columns: 1fr;
                padding: 10px;
            }
        }
    </style>
</head>
<body>
    <div class="cyber-grid"></div>

    <header class="header">
        <h1>      TWO/NGI DEFENSE MATRIX</h1>
        <div class="system-id">Database ID: 7628429 | Professional Cyber
Defense System</div>
        <div class="status-bar">
            <div class="status-item">
                <div class="status-led led-green"></div>
                <span>SYSTEMS ONLINE</span>
            </div>
            <div class="status-item">
                <div class="status-led led-yellow"></div>
                <span>THREAT LEVEL: ELEVATED</span>
            </div>
            <div class="status-item">
                <div class="status-led led-green"></div>
                <span>AI DEFENSE: ACTIVE</span>
            </div>
        </div>
    </header>

    <div class="main-grid">
        <!-- Left Panel - Defense Systems -->
        <div class="panel">
            <div class="panel-header">
```

```
            <h2>      ACTIVE DEFENSES</h2>
        </div>

        <div class="defense-module">
            <div class="module-header">
                <div class="module-name">
                    <span>     </span>
                    <span>Advanced Firewall</span>
                </div>
                <div class="power-switch on" onclick="toggleDefense(this,
'firewall')"></div>
            </div>
            <div style="font-size: 12px; color: #00ccff;">
                Deep Packet Inspection | Geo-IP Blocking | DDoS Protection
            </div>
        </div>

        <div class="defense-module">
            <div class="module-header">
                <div class="module-name">
                    <span>     </span>
                    <span>AI Threat Hunter</span>
                </div>
                <div class="power-switch on" onclick="toggleDefense(this, 'ai-
hunter')"></div>
            </div>
            <div style="font-size: 12px; color: #00ccff;">
                Machine Learning | Behavioral Analysis | Predictive Defense
            </div>
        </div>

        <div class="defense-module">
            <div class="module-header">
                <div class="module-name">
                    <span>     </span>
                    <span>Anti-Malware Engine</span>
                </div>
                <div class="power-switch on" onclick="toggleDefense(this,
'antimalware')"></div>
            </div>
            <div style="font-size: 12px; color: #00ccff;">
                Real-time Scanning | Heuristic Analysis | Zero-Day Detection
            </div>
        </div>

        <div class="defense-module">
            <div class="module-header">
                <div class="module-name">
```

```
                <span>    </span>
                <span>Intrusion Detection</span>
            </div>
            <div class="power-switch on" onclick="toggleDefense(this,
'ids')"></div>
        </div>
        <div style="font-size: 12px; color: #00ccff;">
            Network Monitoring | Anomaly Detection | Auto-Response
        </div>
    </div>

    <div class="defense-module">
        <div class="module-header">
            <div class="module-name">
                <span>    </span>
                <span>Quantum Encryption</span>
            </div>
            <div class="power-switch on" onclick="toggleDefense(this,
'encryption')"></div>
        </div>
        <div style="font-size: 12px; color: #00ccff;">
            AES-256 | RSA-4096 | Quantum-Resistant Algorithms
        </div>
    </div>

    <div class="defense-module">
        <div class="module-header">
            <div class="module-name">
                <span>    </span>
                <span>Proactive Strike</span>
            </div>
            <div class="power-switch on" onclick="toggleDefense(this,
'proactive')"></div>
        </div>
        <div style="font-size: 12px; color: #00ccff;">
            Pre-emptive Neutralization | Counter-Attack | Threat
Elimination
        </div>
    </div>

    <div style="margin-top: 30px;">
        <h4 style="margin-bottom: 15px; color: #ff4444;">
EMERGENCY PROTOCOLS</h4>
        <button class="command-button danger"
onclick="initiateDefcon1()">DEFCON 1</button>
        <button class="command-button warning"
onclick="activateSiegeMode()">SIEGE MODE</button>
        <button class="command-button"
```

```
onclick="fullSystemRestore()">RESTORE ALL</button>
        </div>
      </div>

      <!-- Center Panel - Threat Intelligence -->
      <div class="panel">
        <div class="panel-header">
          <h2>      THREAT INTELLIGENCE CENTER</h2>
        </div>

        <div class="critical-alert">
          <h4 style="color: #ff4444; margin-bottom: 10px;">      ACTIVE
THREAT ASSESSMENT</h4>
          <div>Multiple attack vectors detected | Implementing
countermeasures</div>
        </div>

        <div class="threat-stats">
          <div class="stat-box">
            <div class="stat-value" id="attacks-blocked">15,847</div>
            <div class="stat-label">Attacks Blocked Today</div>
          </div>
          <div class="stat-box">
            <div class="stat-value" id="threats-quarantined">2,394</div>
            <div class="stat-label">Threats Quarantined</div>
          </div>
          <div class="stat-box">
            <div class="stat-value" id="intrusions-stopped">567</div>
            <div class="stat-label">Intrusions Stopped</div>
          </div>
          <div class="stat-box">
            <div class="stat-value" id="malware-eliminated">1,829</div>
            <div class="stat-label">Malware Eliminated</div>
          </div>
        </div>

        <div class="attack-surface">
          <h4 style="margin-bottom: 15px;">      PROTECTED ATTACK
SURFACE</h4>
          <div class="device-protection">
            <div class="device-item"
onclick="scanDeviceType('computer')">
              <span class="device-icon">    </span>
              <div>Computers</div>
              <div style="font-size: 11px; color: #00ff41;">SECURED</
div>
            </div>
            <div class="device-item" onclick="scanDeviceType('mobile')">
```

```
                <span class="device-icon">📱</span>
                <div>Mobile Devices</div>
                <div style="font-size: 11px; color: #00ff41;">PROTECTED</
div>
            </div>
            <div class="device-item" onclick="scanDeviceType('server')">
                <span class="device-icon">🖥️</span>
                <div>Servers</div>
                <div style="font-size: 11px; color: #00ff41;">HARDENED</
div>
            </div>
            <div class="device-item" onclick="scanDeviceType('iot')">
                <span class="device-icon">📡</span>
                <div>IoT Ecosystem</div>
                <div style="font-size: 11px; color: #ffaa00;">MONITORING</
div>
            </div>
            <div class="device-item" onclick="scanDeviceType('network')">
                <span class="device-icon">🌐</span>
                <div>Network Infrastructure</div>
                <div style="font-size: 11px; color: #00ff41;">DEFENDED</
div>
            </div>
            <div class="device-item" onclick="scanDeviceType('cloud')">
                <span class="device-icon">☁️</span>
                <div>Cloud Services</div>
                <div style="font-size: 11px; color: #00ff41;">ENCRYPTED</
div>
            </div>
        </div>
    </div>

    <div style="display: grid; grid-template-columns: repeat(2, 1fr);
gap: 10px; margin: 20px 0;">
        <button class="command-button"
onclick="initiateFullScan()">FULL SPECTRUM SCAN</button>
        <button class="command-button"
onclick="deepThreatHunt()">DEEP THREAT HUNT</button>
        <button class="command-button warning"
onclick="proactiveStrike()">PREEMPTIVE STRIKE</button>
        <button class="command-button danger"
onclick="quarantineProtocol()">MASS QUARANTINE</button>
    </div>

    <div class="progress-bar">
        <div class="progress-fill" id="defense-strength" style="width:
94%;"></div>
    </div>
```

```
        <div style="text-align: center; font-size: 12px; color: #00ccff;">
            Defense Integrity: <span style="color: #00ff41; font-weight:
bold;">94%</span>
        </div>

        <div class="terminal" id="threat-terminal">
[INIT] TWO/NGI Defense Matrix Database 7628429 Online
[AI] Advanced threat hunting algorithms activated
[FIREWALL] Deep packet inspection enabled - monitoring 1,247
connections
[SCAN] Real-time malware detection active across all endpoints
[IDS] Behavioral anomaly detection running - baseline established
[ENCRYPT] Quantum-resistant encryption protocols engaged
[NETWORK] Zero-trust architecture implemented
[PROACTIVE] Counter-attack systems on standby
[STATUS] All defense systems operational - God-tier protection active
[ALERT] 15,847 attacks neutralized in last 24 hours
        </div>
    </div>

    <!-- Right Panel - Configuration & Controls -->
    <div class="panel">
        <div class="panel-header">
            <h2>    SYSTEM CONFIGURATION</h2>
        </div>

        <div class="setting-group">
            <h4 style="margin-bottom: 15px; color: #00ccff;">    DEFENSE
PROTOCOLS</h4>

            <div class="setting-item">
                <label>Defense Level:</label>
                <select id="defense-level" onchange="updateDefenseLevel()"
style="background: #000; color: #00ff41; border: 1px solid #00ff41;
padding: 5px;">
                    <option value="standard">Standard</option>
                    <option value="enhanced">Enhanced</option>
                    <option value="maximum">Maximum</option>
                    <option value="godtier" selected>God-Tier</option>
                </select>
            </div>

            <div class="setting-item">
                <label>Auto-Quarantine:</label>
                <div class="power-switch on" onclick="toggleSetting(this,
'auto-quarantine')"></div>
            </div>
```

```
            <div class="setting-item">
                <label>Proactive Defense:</label>
                <div class="power-switch on" onclick="toggleSetting(this,
'proactive-defense')"></div>
            </div>

            <div class="setting-item">
                <label>AI Threat Prediction:</label>
                <div class="power-switch on" onclick="toggleSetting(this, 'ai-
prediction')"></div>
            </div>

            <div class="setting-item">
                <label>Zero-Day Protection:</label>
                <div class="power-switch on" onclick="toggleSetting(this,
'zero-day')"></div>
            </div>

            <div class="setting-item">
                <label>Counter-Attack Mode:</label>
                <div class="power-switch on" onclick="toggleSetting(this,
'counter-attack')"></div>
            </div>

            <div class="setting-item">
                <label>Deep Web Monitoring:</label>
                <div class="power-switch on" onclick="toggleSetting(this,
'deep-web')"></div>
            </div>
        </div>

        <div class="setting-group">
            <h4 style="margin-bottom: 15px; color: #00ccff;">      DEVICE
PROTECTION</h4>

            <div class="setting-item">
                <label>Smartphones:</label>
                <div class="power-switch on" onclick="toggleDevice(this,
'smartphones')"></div>
            </div>

            <div class="setting-item">
                <label>Computers:</label>
                <div class="power-switch on" onclick="toggleDevice(this,
'computers')"></div>
            </div>

            <div class="setting-item">
```

```
                <label>Servers:</label>
                <div class="power-switch on" onclick="toggleDevice(this,
'servers')"></div>
            </div>

            <div class="setting-item">
                <label>IoT Devices:</label>
                <div class="power-switch on" onclick="toggleDevice(this,
'iot')"></div>
            </div>

            <div class="setting-item">
                <label>Cloud Infrastructure:</label>
                <div class="power-switch on" onclick="toggleDevice(this,
'cloud')"></div>
            </div>
        </div>

        <div class="setting-group">
            <h4 style="margin-bottom: 15px; color: #00ccff;">
ADVANCED OPTIONS</h4>

            <div class="setting-item">
                <label>Scan Frequency:</label>
                <select style="background: #000; color: #00ff41; border: 1px
solid #00ff41; padding: 3px;">
                    <option value="realtime" selected>Real-time</option>
                    <option value="continuous">Continuous</option>
                    <option value="aggressive">Aggressive</option>
                </select>
            </div>

            <div class="setting-item">
                <label>Threat Response:</label>
                <select style="background: #000; color: #00ff41; border: 1px
solid #00ff41; padding: 3px;">
                    <option value="automatic" selected>Automatic</option>
                    <option value="manual">Manual Approval</option>
                    <option value="aggressive">Aggressive Auto</option>
                </select>
            </div>

            <div class="setting-item">
                <label>Logging Level:</label>
                <select style="background: #000; color: #00ff41; border: 1px
solid #00ff41; padding: 3px;">
                    <option value="minimal">Minimal</option>
                    <option value="standard">Standard</option>
```

```
                <option value="verbose" selected>Verbose</option>
                <option value="debug">Debug</option>
              </select>
            </div>
          </div>

          <div style="margin-top: 30px; text-align: center;">
            <h4 style="margin-bottom: 15px; color: #ff4444;">      CRITICAL
CONTROLS</h4>
            <button class="command-button"
onclick="exportConfiguration()">EXPORT CONFIG</button>
            <button class="command-button warning"
onclick="updateAllSystems()">UPDATE ALL</button>
            <button class="command-button danger"
onclick="systemWipe()">SECURE WIPE</button>
          </div>

          <div style="margin-top: 20px;">
            <h4 style="margin-bottom: 10px;">      System Performance</
h4>
            <div>CPU Usage: <span style="color: #00ff41;">23%</span></
div>
            <div>Memory: <span style="color: #00ff41;">4.2GB / 16GB</
span></div>
            <div>Network: <span style="color: #00ff41;">847 Mbps</
span></div>
            <div>Threat Database: <span style="color: #00ff41;">2.4M
signatures</span></div>
          </div>
        </div>
      </div>

      <script>
        // Global defense state
        let defenseState = {
          systems: {
            firewall: true,
            aiHunter: true,
            antimalware: true,
            ids: true,
            encryption: true,
            proactive: true
          },
          counters: {
            attacksBlocked: 15847,
            threatsQuarantined: 2394,
            intrusionsStopped: 567,
            malwareEliminated: 1829
```

```
        },
        logs: [],
        defenseLevel: 'godtier'
    };

    function addThreatLog(message, category = 'SYSTEM', severity =
'INFO') {
        const timestamp = new Date().toLocaleTimeString();
        const logEntry = `[${timestamp}] [${category}] ${message}`;
        defenseState.logs.unshift(logEntry);

        // Keep only last 100 entries
        if (defenseState.logs.length > 100) {
            defenseState.logs = defenseState.logs.slice(0, 100);
        }

        updateThreatDisplay();
    }

    function updateThreatDisplay() {
        const terminal = document.getElementById('threat-terminal');
        terminal.textContent = defenseState.logs.slice(0, 15).join('\n');
        terminal.scrollTop = 0;
    }

    function toggleDefense(element, systemName) {
        element.classList.toggle('on');
        const isActive = element.classList.contains('on');
        defenseState.systems[systemName] = isActive;

        const status = isActive ? 'ACTIVATED' : 'DEACTIVATED';
        addThreatLog(`${systemName.toUpperCase().replace('-', ' ')}
system ${status}`, 'CONFIG', 'WARNING');

        if (!isActive) {
            addThreatLog(`    CRITICAL: ${systemName.toUpperCase()}
defense layer disabled`, 'ALERT', 'CRITICAL');
        }
    }

    function toggleSetting(element, settingName) {
        element.classList.toggle('on');
        const status = element.classList.contains('on') ? 'ENABLED' :
'DISABLED';
        addThreatLog(`Security feature ${settingName.replace('-', '
').toUpperCase()} ${status}`, 'CONFIG');
    }
```

```
    function toggleDevice(element, deviceType) {
        element.classList.toggle('on');
        const status = element.classList.contains('on') ? 'PROTECTED' :
'VULNERABLE';
        addThreatLog(`Device category ${deviceType.toUpperCase()} now
${status}`, 'DEVICE');
    }

    function initiateDefcon1() {
        addThreatLog('     DEFCON 1 INITIATED - MAXIMUM THREAT
RESPONSE     ', 'CRITICAL', 'CRITICAL');
        addThreatLog('All defensive systems at maximum capacity',
'DEFENSE', 'CRITICAL');
        addThreatLog('Implementing scorched earth protocols',
'PROTOCOL', 'CRITICAL');
        addThreatLog('Counter-attack systems fully authorized',
'WEAPON', 'CRITICAL');

        // Visual alert effect
        document.body.style.filter = 'hue-rotate(30deg) brightness(1.2)';
        setTimeout(() => {
            document.body.style.filter = 'none';
        }, 5000);
    }

    function activateSiegeMode() {
        addThreatLog('     SIEGE MODE ACTIVATED - FORTRESS DEFENSE
ENGAGED     ', 'DEFENSE', 'HIGH');
        addThreatLog('All external connections under heavy scrutiny',
'NETWORK', 'HIGH');
        addThreatLog('Proactive threat elimination authorized', 'ATTACK',
'HIGH');
        addThreatLog('System entering maximum defensive posture',
'POSTURE', 'HIGH');

        setTimeout(() => {
            const neutralized = Math.floor(Math.random() * 50) + 20;
            defenseState.counters.attacksBlocked += neutralized;
            document.getElementById('attacks-blocked').textContent =
defenseState.counters.attacksBlocked.toLocaleString();
            addThreatLog(`Siege defense complete: ${neutralized} threats
eliminated`, 'SUCCESS');
        }, 3000);
    }

    function fullSystemRestore() {
        addThreatLog('Initiating complete system restoration...', 'SYSTEM');
        addThreatLog('Resetting all defense systems to optimal
```

```
configuration', 'RESTORE');

        // Reset all switches to on
        document.querySelectorAll('.power-switch').forEach(sw =>
sw.classList.add('on'));

        setTimeout(() => {
            addThreatLog('System restoration complete - all defenses at
100%', 'SUCCESS');
            document.getElementById('defense-strength').style.width =
'100%';
        }, 2000);
    }

    function initiateFullScan() {
        addThreatLog('    INITIATING FULL SPECTRUM THREAT SCAN',
'SCAN');
        addThreatLog('Scanning all protected endpoints and
infrastructure', 'SCAN');
        addThreatLog('AI behavioral analysis active across all systems', 'AI');

        let scanProgress = 0;
        const scanInterval = setInterval(() => {
            scanProgress += Math.random() * 20 + 10;
            addThreatLog(`Scan progress: ${Math.floor(scanProgress)}% -
analyzing threat vectors`, 'SCAN');

            if (scanProgress >= 100) {
                clearInterval(scanInterval);
                const threatsFound = Math.floor(Math.random() * 30) + 10;
                defenseState.counters.threatsQuarantined += threatsFound;
                defenseState.counters.malwareEliminated += threatsFound;

                document.getElementById('threats-quarantined').textContent
= defenseState.counters.threatsQuarantined.toLocaleString();
                document.getElementById('malware-eliminated').textContent
= defenseState.counters.malwareEliminated.toLocaleString();

                addThreatLog(`    FULL SCAN COMPLETE: ${threatsFound}
threats neutralized and quarantined`, 'SUCCESS');
                addThreatLog('All identified malware eliminated from
systems', 'CLEANUP');
            }
        }, 1200);
    }

    function deepThreatHunt() {
        addThreatLog('    INITIATING DEEP THREAT HUNTING
```

```
OPERATION', 'HUNT');
        addThreatLog('AI algorithms analyzing behavioral patterns and
anomalies', 'AI');
        addThreatLog('Cross-referencing with global threat intelligence
databases', 'INTEL');
        addThreatLog('Hunting for advanced persistent threats and zero-
days', 'APT');

        setTimeout(() => {
          const aptsFound = Math.floor(Math.random() * 8) + 2;
          const zerodays = Math.floor(Math.random() * 3) + 1;

          addThreatLog(`Deep hunt complete: ${aptsFound} APTs and $
{zerodays} zero-day exploits detected`, 'DISCOVERY');
          addThreatLog('Implementing custom signatures and behavioral
blocks', 'SIGNATURE');
          addThreatLog('Threat intelligence database updated with new
IOCs', 'IOC');

          defenseState.counters.intrusionsStopped += aptsFound;
          document.getElementById('intrusions-stopped').textContent =
defenseState.counters.intrusionsStopped.toLocaleString();
        }, 4500);
      }

      function proactiveStrike() {
        addThreatLog('    PROACTIVE STRIKE AUTHORIZATION
CONFIRMED', 'STRIKE', 'HIGH');
        addThreatLog('Identifying threat sources for pre-emptive
neutralization', 'TARGET', 'HIGH');
        addThreatLog('Launching counter-attack algorithms', 'ATTACK',
'HIGH');
        addThreatLog('Deploying offensive security measures', 'OFFENSE',
'HIGH');

        setTimeout(() => {
          const sourcesNeutralized = Math.floor(Math.random() * 15) + 5;
          defenseState.counters.attacksBlocked += sourcesNeutralized * 3;
          document.getElementById('attacks-blocked').textContent =
defenseState.counters.attacksBlocked.toLocaleString();

          addThreatLog(`    STRIKE COMPLETE: ${sourcesNeutralized}
threat sources neutralized`, 'SUCCESS');
          addThreatLog('Pre-emptive defense successful - attack vectors
eliminated', 'ELIMINATION');
          addThreatLog('Threat landscape significantly reduced',
'TACTICAL');
        }, 3500);
```

```
    }

    function quarantineProtocol() {
        addThreatLog('     MASS QUARANTINE PROTOCOL INITIATED',
'QUARANTINE', 'HIGH');
        addThreatLog('Isolating all suspicious files, processes, and
network connections', 'ISOLATION');
        addThreatLog('Emergency containment procedures active',
'CONTAINMENT');

        setTimeout(() => {
            const quarantined = Math.floor(Math.random() * 100) + 50;
            defenseState.counters.threatsQuarantined += quarantined;
            document.getElementById('threats-quarantined').textContent =
defenseState.counters.threatsQuarantined.toLocaleString();

            addThreatLog(`     MASS QUARANTINE COMPLETE: $
{quarantined} items secured`, 'SUCCESS');
            addThreatLog('All identified threats isolated and neutralized',
'SECURE');
        }, 2500);
    }

    function scanDeviceType(deviceType) {
        addThreatLog(`     Initiating comprehensive $
{deviceType.toUpperCase()} security scan`, 'DEVICE');
        addThreatLog(`Analyzing ${deviceType} infrastructure for
vulnerabilities`, 'VULN');

        setTimeout(() => {
            const threatsFound = Math.floor(Math.random() * 8);
            if (threatsFound > 0) {
                addThreatLog(`${deviceType.toUpperCase()} scan: $
{threatsFound} threats detected and neutralized`, 'DEVICE');
                defenseState.counters.attacksBlocked += threatsFound;
                document.getElementById('attacks-blocked').textContent =
defenseState.counters.attacksBlocked.toLocaleString();
            } else {
                addThreatLog(`${deviceType.toUpperCase()} scan: All systems
clean - no threats detected`, 'CLEAN');
            }
            addThreatLog(`${deviceType.toUpperCase()} infrastructure
security verified`, 'VERIFY');
        }, 2000);
    }

    function updateDefenseLevel() {
        const level = document.getElementById('defense-level').value;
```

```
        defenseState.defenseLevel = level;
        addThreatLog(`Defense level updated to: ${level.toUpperCase()}`,
'CONFIG');

        switch(level) {
            case 'godtier':
                addThreatLog('     GOD-TIER DEFENSE ACTIVATED -
MAXIMUM PROTECTION', 'UPGRADE', 'HIGH');
                addThreatLog('All advanced AI systems online -  predictive
threat neutralization active', 'AI', 'HIGH');
                addThreatLog('Quantum-resistant encryption and proactive
strike capabilities enabled', 'QUANTUM', 'HIGH');
                document.getElementById('defense-strength').style.width =
'100%';
                break;
            case 'maximum':
                addThreatLog('Maximum security protocols engaged',
'UPGRADE');
                document.getElementById('defense-strength').style.width =
'85%';
                break;
            case 'enhanced':
                addThreatLog('Enhanced protection mode activated',
'UPGRADE');
                document.getElementById('defense-strength').style.width =
'70%';
                break;
            case 'standard':
                addThreatLog('Standard defense protocols active',
'STANDARD');
                document.getElementById('defense-strength').style.width =
'50%';
                break;
        }
    }

    function exportConfiguration() {
        addThreatLog('     Exporting current security configuration...',
'EXPORT');
        setTimeout(() => {
            addThreatLog('Security configuration exported to secure
encrypted backup', 'SUCCESS');
            addThreatLog('Configuration includes all defense rules and
threat signatures', 'BACKUP');
        }, 1500);
    }

    function updateAllSystems() {
```

```
        addThreatLog('      INITIATING COMPREHENSIVE SYSTEM UPDATE',
'UPDATE');
        addThreatLog('Downloading latest threat signatures and security
patches', 'DOWNLOAD');
        addThreatLog('Updating AI models with newest attack patterns',
'AI-UPDATE');

        setTimeout(() => {
          addThreatLog('System update complete: 25,847 new threat
signatures installed', 'SUCCESS');
          addThreatLog('AI behavioral models updated with latest threat
intelligence', 'AI-SUCCESS');
          addThreatLog('All defense systems operating at peak efficiency',
'OPTIMAL');
        }, 3000);
      }

      function systemWipe() {
        if (confirm('      WARNING: This will perform a secure wipe of all
threat data and reset to factory defaults. Continue?')) {
          addThreatLog('      SECURE SYSTEM WIPE INITIATED', 'WIPE',
'CRITICAL');
          addThreatLog('Performing 7-pass DoD secure erase of threat
databases', 'ERASE', 'CRITICAL');
          addThreatLog('Resetting all configurations to factory defaults',
'RESET', 'CRITICAL');

          setTimeout(() => {
            addThreatLog('Secure wipe complete -  system restored to
clean state', 'SUCCESS');
            addThreatLog('All threat traces eliminated -  fresh defense
installation', 'CLEAN');
            // Reset counters
            Object.keys(defenseState.counters).forEach(key =>
defenseState.counters[key] = 0);
            document.querySelectorAll('.stat-value').forEach(el =>
el.textContent = '0');
          }, 4000);
        }
      }

      // Auto-update system with realistic threat simulation
      setInterval(() => {
        const eventChance = Math.random();

        if (eventChance < 0.15) {
          const newThreats = Math.floor(Math.random() * 5) + 1;
          defenseState.counters.attacksBlocked += newThreats;
```

```
            document.getElementById('attacks-blocked').textContent =
defenseState.counters.attacksBlocked.toLocaleString();
            addThreatLog(`${newThreats} incoming attacks automatically
blocked`, 'AUTO-DEFENSE');
        }

        if (eventChance > 0.85 && eventChance < 0.9) {
            addThreatLog('Routine security health check completed - all
systems optimal', 'HEALTH');
        }

        if (eventChance > 0.95) {
            const threatType = ['botnet C&C', 'phishing attempt', 'malware
payload', 'intrusion probe'][Math.floor(Math.random() * 4)];
            addThreatLog(`Advanced ${threatType} detected and
neutralized by AI systems`, 'AI-DEFENSE');
        }
    }, 8000);

    // Initialize system with startup sequence
    setTimeout(() => {
        addThreatLog('     TWO/NGI Defense Matrix fully initialized and
operational', 'INIT');
        addThreatLog('All protection modules active - God-tier security
engaged', 'STATUS');
        addThreatLog('Proactive threat hunting and counter-attack
systems online', 'READY');
    }, 1000);

    // Simulate real-time threat activity
    setTimeout(() => {
        addThreatLog('Real-time threat feed active - monitoring global
attack patterns', 'FEED');
    }, 3000);
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta 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>
</body>
</html>
 s <!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <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>
</head>
<body>
  <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 &
www.bitcoinayt.world â€” client- side demo</div>
      </div>
      <div class="small">Session: <span id="sessionId" class="badge">ngi-
sess- 0001</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>

    <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 & 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 & 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 & Moderation Log</div>
      <div id="logBox" class="simBox" style="height:120px">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 & 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 & 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>
</body>
</html>
 s <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<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>
</head>
<body>

<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('translate')">Translate
Thoughts</div>
    <div class="taskBtn"
onclick="performTask('summarize')">Summarize</div>
    <div class="taskBtn" onclick="performTask('analyze')">Analyze
Emotion</div>
    <div class="taskBtn" onclick="performTask('database')">Query
Database 82698</div>
    <div class="taskBtn" onclick="performTask('llama')">LLaMA
Response</div>
    <div class="taskBtn" onclick="performTask('sevenForms')">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>

</body>
</html>
 s <!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <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:tr
anslateX(-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>
</head>
<body>
  <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>
```