```
size="10" font-weight="bold">HIPP</text>

            <!-- Neural pathways -->
            <path class="neural-pathway" d="M210 170 Q250 190 280
210"></path>
            <path class="neural-pathway" d="M310 220 Q330 200 350
180"></path>
            <path class="neural-pathway" d="M300 240 Q320 230 340
210"></path>
        </svg>

        <div class="brain-waves">
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
            <div class="wave"></div>
        </div>
    </div>

    <!-- Region Information Panels -->
    <div id="prefrontal" class="info-panel">
        <h3>Prefrontal Cortex (PFC)</h3>
        <p><strong>Function:</strong> Executive control, working
memory, planning</p>
        <p><strong>Role in Decision-Making:</strong> Evaluates
options, considers consequences, implements decisions</p>
        <p><strong>Key Processes:</strong> Inhibitory control,
cognitive flexibility, goal management</p>
    </div>

    <div id="amygdala" class="info-panel">
        <h3>Amygdala</h3>
        <p><strong>Function:</strong> Emotion processing, threat
detection, social predictions</p>
        <p><strong>Role in Decision-Making:</strong> Processes
emotional significance, predicts others' choices</p>
        <p><strong>Key Processes:</strong> Fear conditioning, reward
prediction, social cognition</p>
    </div>

    <div id="cingulate" class="info-panel">
        <h3>Posterior Cingulate Cortex (PCC)</h3>
        <p><strong>Function:</strong> Self-referential thinking,
memory integration</p>
        <p><strong>Role in Decision-Making:</strong> Integrates past
experiences with current decisions</p>
        <p><strong>Key Processes:</strong> Autobiographical memory,
```

```
value assessment, attention</p>
        </div>

        <div id="hippocampus" class="info-panel">
            <h3>Hippocampus</h3>
            <p><strong>Function:</strong> Memory formation, spatial
navigation</p>
            <p><strong>Role in Decision-Making:</strong> Provides context
from past experiences</p>
            <p><strong>Key Processes:</strong> Memory consolidation,
pattern completion, future simulation</p>
        </div>

        <div class="decision-steps">
            <div class="step-card" onclick="simulateStep(1)">
                <div class="step-number">1</div>
                <h3>Information Gathering</h3>
                <p>Brain collects sensory input and retrieves relevant
memories</p>
            </div>

            <div class="step-card" onclick="simulateStep(2)">
                <div class="step-number">2</div>
                <h3>Option Generation</h3>
                <p>Multiple possible actions are considered simultaneously</p>
            </div>

            <div class="step-card" onclick="simulateStep(3)">
                <div class="step-number">3</div>
                <h3>Value Assessment</h3>
                <p>Each option is evaluated for potential outcomes and
rewards</p>
            </div>

            <div class="step-card" onclick="simulateStep(4)">
                <div class="step-number">4</div>
                <h3>Selection Process</h3>
                <p>Best option is selected based on integrated assessment</
p>
            </div>

            <div class="step-card" onclick="simulateStep(5)">
                <div class="step-number">5</div>
                <h3>Action Execution</h3>
                <p>Motor cortex implements the chosen decision</p>
            </div>

            <div class="step-card" onclick="simulateStep(6)">
```

```
            <div class="step-number">6</div>
            <h3>Outcome Learning</h3>
            <p>Results are evaluated and stored for future decisions</p>
        </div>
    </div>

    <div class="interactive-demo">
        <h3>Decision-Making Simulator</h3>
        <p>Click the scenario buttons to see how the brain processes
different types of decisions:</p>

        <button class="demo-button"
onclick="simulateDecision(&#39;simple&#39;)">Simple Choice</button>
        <button class="demo-button"
onclick="simulateDecision(&#39;complex&#39;)">Complex Problem</
button>
        <button class="demo-button"
onclick="simulateDecision(&#39;social&#39;)">Social Decision</button>
        <button class="demo-button"
onclick="simulateDecision(&#39;moral&#39;)">Moral Dilemma</button>

        <div id="simulation-output" class="thinking-process">
            Click a scenario to see neural processing...
        </div>
    </div>

    <div class="question-simulator">
        <h3>Brain Question Processing</h3>
        <p>Enter a question to see how the brain might process it
through different evaluation stages:</p>
        <input type="text" class="question-input" id="questionInput"
placeholder="Enter your question here...">
        <button class="demo-button"
onclick="processQuestion()">Process Question</button>

        <div class="timeline" id="questionTimeline">
            <!-- Timeline items will be generated here -->
        </div>
    </div>

    <div class="brain-container">
        <h3>Real Neuroscience vs. Speculative Claims</h3>
        <div style="display: grid; grid-template-columns: 1fr 1fr; gap:
20px; margin-top: 20px;">
            <div style="background: #d5f4e6; padding: 20px; border-radius:
10px;">
                <h4 style="color: #27ae60;">Established Science</h4>
                <ul style="margin-top: 10px; line-height: 1.6;">
```

```
                <li>fMRI can detect brain activity patterns</li>
                <li>EEG measures electrical brain signals</li>
                <li>Basic brain-computer interfaces exist</li>
                <li>Decision-making involves multiple brain regions</li>
                <li>Neural networks process information in parallel</li>
            </ul>
        </div>
        <div style="background: #ffeaa7; padding: 20px; border-radius:
10px;">
            <h4 style="color: #f39c12;">Current Limitations</h4>
            <ul style="margin-top: 10px; line-height: 1.6;">
                <li>Cannot read specific thoughts</li>
                <li>No "mind control" technology exists</li>
                <li>Animal communication is limited</li>
                <li>No direct brain-to-brain transfer</li>
                <li>DNA cannot be "calculated" from appearance</li>
            </ul>
        </div>
    </div>
</div>

<script>
    // Real neuroscience-based decision processing questions
    const decisionQuestions = [
        "What information is available?",
        "What are my options?",
        "What are the potential outcomes?",
        "What are the risks and benefits?",
        "What do I value most?",
        "What does past experience suggest?",
        "What would others expect?",
        "What are the time constraints?",
        "What are the resource requirements?",
        "What is my emotional response?",
        "What is the likely success rate?",
        "What are the unintended consequences?"
    ];

    const brainRegions = {
        prefrontal: {
            name: "Prefrontal Cortex",
            function: "Executive control and planning",
            processes: ["Working memory", "Inhibitory control", "Cognitive
flexibility"]
        },
        amygdala: {
            name: "Amygdala",
```

```
            function: "Emotion and threat processing",
            processes: ["Fear detection", "Emotional memories", "Social
predictions"]
        },
        cingulate: {
            name: "Posterior Cingulate Cortex",
            function: "Self-referential processing",
            processes: ["Autobiographical memory", "Value assessment",
"Attention control"]
        },
        hippocampus: {
            name: "Hippocampus",
            function: "Memory formation and retrieval",
            processes: ["Episodic memory", "Spatial navigation", "Future
simulation"]
        }
    };

    function showRegionInfo(region) {
        // Hide all panels
        document.querySelectorAll('.info-panel').forEach(panel => {
            panel.classList.remove('active');
        });

        // Show selected panel
        document.getElementById(region).classList.add('active');

        // Highlight brain region
        document.querySelectorAll('.brain-region').forEach(r => {
            r.style.stroke = '#667eea';
            r.style.strokeWidth = '2';
        });

        document.querySelector(`[data-region="${region}"]`).style.stroke =
'#e74c3c';
        document.querySelector(`[data-region="$
{region}"]`).style.strokeWidth = '4';
    }

    function simulateDecision(type) {
        const scenarios = {
            simple: {
                scenario: "Choosing between coffee or tea",
                process: [
                    "Sensory input: See coffee and tea options",
                    "Memory retrieval: Past preferences accessed",
                    "Value assessment: Taste, energy, mood considered",
                    "Quick decision: Coffee selected based on caffeine need",
```

```
                "Motor execution: Reach for coffee",
                "Outcome monitoring: Satisfaction with choice"
            ]
        },
        complex: {
            scenario: "Deciding on a career change",
            process: [
                "Information gathering: Current job satisfaction, market
research",
                "Option generation: Multiple career paths considered",
                "Risk analysis: Financial security vs. personal fulfillment",
                "Future simulation: Imagining different life outcomes",
                "Social consideration: Family impact, peer opinions",
                "Values integration: Aligning choice with core beliefs",
                "Decision commitment: Weighing all factors",
                "Action planning: Steps to implement change"
            ]
        },
        social: {
            scenario: "Responding to a friend's request for help",
            process: [
                "Social signal processing: Friend's emotional state detected",
                "Relationship assessment: Importance of friendship
evaluated",
                "Capability analysis: Own ability to help considered",
                "Competing priorities: Other commitments weighed",
                "Empathy activation: Friend's perspective considered",
                "Reciprocity calculation: Past help given/received",
                "Decision: Agreement to help based on social bonds"
            ]
        },
        moral: {
            scenario: "Finding a wallet with money",
            process: [
                "Moral recognition: Ethical dimension identified",
                "Rule retrieval: Social norms about honesty accessed",
                "Consequence prediction: Outcomes of keeping vs.
returning",
                "Empathy activation: Owner's potential distress considered",
                "Value conflict: Personal gain vs. moral principles",
                "Social reputation: What others would think",
                "Moral decision: Return wallet based on ethical values"
            ]
        }
    };

    const scenario = scenarios[type];
    const output = document.getElementById('simulation-output');
```

```
        output.innerHTML = `<strong>Scenario:</strong> ${scenario.scenario}<br><br>`;

        scenario.process.forEach((step, index) => {
            setTimeout(() => {
                output.innerHTML += `<span style="color: #f39c12;">[${(index + 1) * 200}ms]</span> ${step}<br>`;
                output.scrollTop = output.scrollHeight;
            }, index * 800);
        });
    }

    function processQuestion() {
        const question = document.getElementById('questionInput').value;
        if (!question.trim()) return;

        const timeline = document.getElementById('questionTimeline');
        timeline.innerHTML = '';

        // Generate processing stages based on the question
        const stages = [
            `Question received: "${question}"`,
            "Semantic analysis: Breaking down meaning",
            "Memory search: Accessing relevant information",
            "Pattern matching: Finding similar past experiences",
            "Option generation: Creating possible answers",
            "Probability assessment: Evaluating likelihood of outcomes",
            "Value alignment: Checking against personal beliefs",
            "Confidence calculation: Assessing certainty level",
            "Response formulation: Constructing answer",
            "Output preparation: Ready to respond"
        ];

        stages.forEach((stage, index) => {
            const timelineItem = document.createElement('div');
            timelineItem.className = 'timeline-item';
            timelineItem.innerHTML = `
                <strong>Stage ${index + 1}:</strong> ${stage}<br>
                <small>Processing time: ${(index + 1) * 150}ms</small>
            `;
            timeline.appendChild(timelineItem);

            setTimeout(() => {
                timelineItem.classList.add('active');
            }, index * 600);
        });
    }
```

```
        function simulateStep(stepNumber) {
            const steps = {
                1: "Information Gathering: Sensory inputs processed, memories
activated",
                2: "Option Generation: Multiple pathways explored
simultaneously",
                3: "Value Assessment: Prefrontal cortex weighs pros and cons",
                4: "Selection Process: Winner-take-all competition between
options",
                5: "Action Execution: Motor commands sent to implement
decision",
                6: "Outcome Learning: Results fed back to update future
decisions"
            };

            alert(`Step ${stepNumber}: ${steps[stepNumber]}`);
        }

        // Add some dynamic brain activity
        setInterval(() => {
            const regions = document.querySelectorAll('.brain-region');
            const randomRegion = regions[Math.floor(Math.random() *
regions.length)];

            randomRegion.style.filter = 'brightness(1.3)';
            setTimeout(() => {
                randomRegion.style.filter = 'brightness(1)';
            }, 300);
        }, 2000);

        // Initialize
        document.addEventListener('DOMContentLoaded', function() {
            // Show default info
            setTimeout(() => {
                showRegionInfo('prefrontal');
            }, 1000);
        });
    </script>


s



    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Real Brain Action Potentials - Interactive Guide</title>
    <style>
```

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

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background: linear-gradient(135deg, #2c3e50, #3498db);
    color: #333;
    min-height: 100vh;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
}

.header {
    text-align: center;
    color: white;
    margin-bottom: 30px;
}

.neuron-container {
    background: white;
    border-radius: 15px;
    padding: 30px;
    margin: 20px 0;
    box-shadow: 0 10px 30px rgba(0,0,0,0.1);
}

.neuron-svg {
    width: 100%;
    max-width: 800px;
    margin: 0 auto;
    display: block;
}

.action-potential-graph {
    background: #f8f9fa;
    padding: 20px;
    border-radius: 10px;
    margin: 20px 0;
}

.voltage-line {
```

```
    stroke: #e74c3c;
    stroke-width: 3;
    fill: none;
    animation: drawLine 3s ease-in-out infinite;
}

@keyframes drawLine {
    0% { stroke-dasharray: 0, 1000; }
    50% { stroke-dasharray: 500, 1000; }
    100% { stroke-dasharray: 1000, 1000; }
}

.ion-channel {
    cursor: pointer;
    transition: all 0.3s ease;
}

.ion-channel:hover {
    transform: scale(1.1);
    filter: brightness(1.2);
}

.info-panel {
    background: #ecf0f1;
    padding: 20px;
    border-radius: 10px;
    margin: 20px 0;
    display: none;
}

.info-panel.active {
    display: block;
    animation: slideIn 0.5s ease;
}

@keyframes slideIn {
    from { opacity: 0; transform: translateY(20px); }
    to { opacity: 1; transform: translateY(0); }
}

.phases-container {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(250px, 1fr));
    gap: 20px;
    margin: 30px 0;
}

.phase-card {
```

```
    background: linear-gradient(135deg, #667eea, #764ba2);
    color: white;
    padding: 20px;
    border-radius: 10px;
    cursor: pointer;
    transition: all 0.3s ease;
    text-align: center;
}

.phase-card:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 25px rgba(0,0,0,0.2);
}

.simulator {
    background: #2c3e50;
    color: white;
    padding: 25px;
    border-radius: 15px;
    margin: 30px 0;
}

.control-button {
    background: #e74c3c;
    color: white;
    border: none;
    padding: 12px 25px;
    border-radius: 25px;
    cursor: pointer;
    margin: 10px;
    transition: all 0.3s ease;
}

.control-button:hover {
    background: #c0392b;
    transform: scale(1.05);
}

.membrane-visual {
    background: #34495e;
    height: 200px;
    border-radius: 10px;
    position: relative;
    overflow: hidden;
    margin: 20px 0;
}

.ion {
```

```
        width: 20px;
        height: 20px;
        border-radius: 50%;
        position: absolute;
        animation: drift 4s linear infinite;
    }

    .sodium { background: #f39c12; }
    .potassium { background: #9b59b6; }
    .chloride { background: #27ae60; }

    @keyframes drift {
        0% { transform: translateX(-20px); }
        100% { transform: translateX(calc(100vw + 20px)); }
    }

    .comparison-table {
        background: white;
        border-radius: 10px;
        overflow: hidden;
        margin: 20px 0;
    }

    .comparison-table table {
        width: 100%;
        border-collapse: collapse;
    }

    .comparison-table th,
    .comparison-table td {
        padding: 15px;
        text-align: left;
        border-bottom: 1px solid #ddd;
    }

    .comparison-table th {
        background: #f8f9fa;
        font-weight: bold;
    }

    .fact { background: #d5f4e6; color: #27ae60; }
    .fiction { background: #ffeaa7; color: #f39c12; }
</style>


<div class="container">
    <header class="header">
        <h1>Real Brain Action Potentials</h1>
```

```
        <p>Understanding How Neurons Actually Communicate</p>
    </header>

    <div class="neuron-container">
        <h2>Neuron Structure &amp;amp; Action Potential</h2>
        <svg class="neuron-svg" viewBox="0 0 800 400" xmlns="http://
www.w3.org/2000/svg">
            <!-- Neuron cell body -->
            <circle cx="150" cy="200" r="50" fill="#f8c471" stroke="#e67e22"
stroke-width="2"></circle>
            <text x="150" y="205" text-anchor="middle" font-size="12" font-
weight="bold">Cell Body</text>

            <!-- Dendrites -->
            <path d="M100 160 Q80 140 70 130" stroke="#52c41a" stroke-
width="3" fill="none"></path>
            <path d="M110 170 Q90 150 85 145" stroke="#52c41a" stroke-
width="3" fill="none"></path>
            <path d="M100 240 Q80 260 70 270" stroke="#52c41a" stroke-
width="3" fill="none"></path>
            <path d="M110 230 Q90 250 85 255" stroke="#52c41a" stroke-
width="3" fill="none"></path>
            <text x="70" y="125" font-size="10" fill="#52c41a">Dendrites</
text>

            <!-- Axon -->
            <rect x="200" y="190" width="400" height="20" fill="#74b9ff"
stroke="#0984e3" stroke-width="2"></rect>
            <text x="400" y="215" text-anchor="middle" font-size="12"
fill="#0984e3">Axon</text>

            <!-- Myelin sheaths -->
            <rect x="220" y="185" width="40" height="30" fill="#fdcb6e"
stroke="#e17055" stroke-width="1"></rect>
            <rect x="280" y="185" width="40" height="30" fill="#fdcb6e"
stroke="#e17055" stroke-width="1"></rect>
            <rect x="340" y="185" width="40" height="30" fill="#fdcb6e"
stroke="#e17055" stroke-width="1"></rect>
            <rect x="400" y="185" width="40" height="30" fill="#fdcb6e"
stroke="#e17055" stroke-width="1"></rect>
            <rect x="460" y="185" width="40" height="30" fill="#fdcb6e"
stroke="#e17055" stroke-width="1"></rect>
            <text x="350" y="175" font-size="10" fill="#e17055">Myelin
Sheath</text>

            <!-- Nodes of Ranvier -->
            <circle cx="270" cy="200" r="3" fill="#e74c3c" class="ion-channel"
onclick="showInfo(&amp;#39;node&amp;#39;)"></circle>
```

```
            <circle cx="390" cy="200" r="3" fill="#e74c3c" class="ion-channel"
onclick="showInfo(&#39;node&#39;)"></circle>
            <circle cx="450" cy="200" r="3" fill="#e74c3c" class="ion-channel"
onclick="showInfo(&#39;node&#39;)"></circle>

            <!-- Synapse -->
            <circle cx="650" cy="200" r="25" fill="#a29bfe" stroke="#6c5ce7"
stroke-width="2"></circle>
            <text x="650" y="205" text-anchor="middle" font-size="10" font-
weight="bold">Synapse</text>

            <!-- Action potential wave -->
            <circle cx="300" cy="200" r="8" fill="#e74c3c" opacity="0.8"
class="action-potential-wave" style="opacity: 0.8;">
                <animate attributeName="cx" values="200;600;200" dur="3s"
repeatCount="indefinite"></animate>
                <animate attributeName="opacity" values="0;1;0" dur="3s"
repeatCount="indefinite"></animate>
            </circle>
        </svg>
    </div>

    <div class="action-potential-graph">
        <h3>Action Potential Voltage Graph</h3>
        <svg width="100%" height="300" viewBox="0 0 600 300">
            <!-- Axes -->
            <line x1="50" y1="250" x2="550" y2="250" stroke="#333" stroke-
width="2"></line>
            <line x1="50" y1="50" x2="50" y2="250" stroke="#333" stroke-
width="2"></line>

            <!-- Labels -->
            <text x="300" y="280" text-anchor="middle" font-size="14">Time
(ms)</text>
            <text x="20" y="150" text-anchor="middle" font-size="14"
transform="rotate(-90 20 150)">Voltage (mV)</text>

            <!-- Voltage markers -->
            <line x1="45" y1="150" x2="55" y2="150" stroke="#666"></line>
            <text x="40" y="155" text-anchor="end" font-size="10">0</text>
            <line x1="45" y1="100" x2="55" y2="100" stroke="#666"></line>
            <text x="40" y="105" text-anchor="end" font-size="10">+40</
text>
            <line x1="45" y1="200" x2="55" y2="200" stroke="#666"></line>
            <text x="40" y="205" text-anchor="end" font-size="10">-70</
text>

            <!-- Action potential curve -->
```

```
            <path class="voltage-line" d="M50 200 L150 200 L200 100 L250
180 L300 220 L400 200 L550 200"></path>

            <!-- Phase labels -->
            <text x="100" y="190" font-size="10" fill="#666">Resting</text>
            <text x="180" y="90" font-size="10" fill="#666">Depolarization</
text>
            <text x="270" y="170" font-size="10" fill="#666">Repolarization</
text>
            <text x="350" y="210" font-size="10"
fill="#666">Hyperpolarization</text>
        </svg>
    </div>

    <div class="phases-container">
        <div class="phase-card"
onclick="simulatePhase(&#39;resting&#39;)">
            <h3>1. Resting Potential</h3>
            <p>-70mV: Na+/K+ pump maintains voltage</p>
        </div>

        <div class="phase-card"
onclick="simulatePhase(&#39;threshold&#39;)">
            <h3>2. Threshold</h3>
            <p>-55mV: Voltage-gated Na+ channels open</p>
        </div>

        <div class="phase-card"
onclick="simulatePhase(&#39;depolarization&#39;)">
            <h3>3. Depolarization</h3>
            <p>+40mV: Rapid Na+ influx</p>
        </div>

        <div class="phase-card"
onclick="simulatePhase(&#39;repolarization&#39;)">
            <h3>4. Repolarization</h3>
            <p>K+ channels open, Na+ channels close</p>
        </div>

        <div class="phase-card"
onclick="simulatePhase(&#39;hyperpolarization&#39;)">
            <h3>5. Hyperpolarization</h3>
            <p>-90mV: K+ channels stay open briefly</p>
        </div>

        <div class="phase-card"
onclick="simulatePhase(&#39;recovery&#39;)">
            <h3>6. Recovery</h3>
```

```
            <p>Return to resting potential</p>
        </div>
    </div>

    <div class="simulator">
        <h3>Action Potential Simulator</h3>
        <p>Click to trigger different types of neural activity:</p>

        <button class="control-button"
onclick="simulateStimulus(&#39;weak&#39;)">Weak Stimulus</button>
        <button class="control-button"
onclick="simulateStimulus(&#39;threshold&#39;)">Threshold Stimulus</
button>
        <button class="control-button"
onclick="simulateStimulus(&#39;strong&#39;)">Strong Stimulus</
button>
        <button class="control-button"
onclick="simulateStimulus(&#39;train&#39;)">Train of Impulses</button>

        <div class="membrane-visual" id="membraneVisual">
            <!-- Ions will be generated here -->
        </div>

        <div id="simulationOutput" style="background: #34495e; padding:
15px; border-radius: 5px; margin-top: 15px; font-family: monospace;
font-size: 12px; min-height: 100px;">
            Click a stimulus button to see neural activity...
        </div>
    </div>

    <div class="comparison-table">
        <h3 style="padding: 20px; margin: 0;">Real Science vs. Fictional
Claims</h3>
        <table>
            <thead>
                <tr>
                    <th>Topic</th>
                    <th>Real Neuroscience</th>
                    <th>Fictional Claims</th>
                </tr>
            </thead>
            <tbody>
                <tr>
                    <td><strong>Action Potentials</strong></td>
                    <td class="fact">Electrical signals (-70mV to +40mV) that
travel along neurons</td>
                    <td class="fiction">Magical "brain codes" that control
thoughts</td>
```

```
            </tr>
            <tr>
              <td><strong>Brain Communication</strong></td>
              <td class="fact">Chemical synapses using
neurotransmitters</td>
              <td class="fiction">Electromagnetic wave telepathy</td>
            </tr>
            <tr>
              <td><strong>Neural Networks</strong></td>
              <td class="fact">Complex networks of ~86 billion neurons</
td>
              <td class="fiction">Simple numerical codes for everything</
td>
            </tr>
            <tr>
              <td><strong>Brain Reading</strong></td>
              <td class="fact">fMRI/EEG can detect broad activity
patterns</td>
              <td class="fiction">Perfect thought reading devices</td>
            </tr>
            <tr>
              <td><strong>Frequencies</strong></td>
              <td class="fact">Brainwaves: Delta (0.5-4Hz), Alpha
(8-13Hz), etc.</td>
              <td class="fiction">Special "738Hz frequency of creation"</
td>
            </tr>
          </tbody>
        </table>
      </div>

      <div class="neuron-container">
        <h3>What Action Potentials Actually Do</h3>
        <div style="display: grid; grid-template-columns: repeat(auto-fit,
minmax(300px, 1fr)); gap: 20px;">
          <div>
            <h4>Motor Neurons</h4>
            <ul>
              <li>Control muscle contractions</li>
              <li>Signal from brain to muscles</li>
              <li>Example: Moving your hand</li>
            </ul>
          </div>
          <div>
            <h4>Sensory Neurons</h4>
            <ul>
              <li>Detect environmental stimuli</li>
              <li>Send signals to brain</li>
```

```
                <li>Example: Feeling touch, pain</li>
            </ul>
        </div>
        <div>
            <h4>Interneurons</h4>
            <ul>
                <li>Process information in brain</li>
                <li>Connect other neurons</li>
                <li>Example: Memory formation</li>
            </ul>
        </div>
    </div>
</div>

<!-- Info panels for different components -->
<div id="node" class="info-panel active">
    <h3>Nodes of Ranvier</h3>
    <p><strong>Function:</strong> Gaps in myelin sheath where
action potentials are regenerated</p>
    <p><strong>Mechanism:</strong> High concentration of
voltage-gated sodium channels</p>
    <p><strong>Purpose:</strong> Allows rapid, efficient signal
transmission via saltatory conduction</p>
</div>
</div>

<script>
    function showInfo(type) {
        // Hide all info panels
        document.querySelectorAll('.info-panel').forEach(panel => {
            panel.classList.remove('active');
        });

        // Show selected panel
        document.getElementById(type).classList.add('active');
    }

    function simulatePhase(phase) {
        const phases = {
            resting: {
                description: "Resting Potential: -70mV",
                details: "Na+/K+ pump maintains 3:2 ratio, membrane
polarized",
                voltage: -70
            },
            threshold: {
                description: "Threshold: -55mV reached",
                details: "Voltage-gated Na+ channels begin to open",
```

```
            voltage: - 55
          },
          depolarization: {
            description: "Depolarization: Rapid rise to +40mV",
            details: "Na+ channels fully open, positive feedback loop",
            voltage: 40
          },
          repolarization: {
            description: "Repolarization: Na+ channels close",
            details: "K+ channels open, membrane becomes negative
again",
            voltage: - 70
          },
          hyperpolarization: {
            description: "Hyperpolarization: Brief undershoot to - 90mV",
            details: "K+ channels slow to close, membrane extra negative",
            voltage: - 90
          },
          recovery: {
            description: "Recovery: Return to resting potential",
            details: "Na+/K+ pump restores ion gradients",
            voltage: - 70
          }
        };

        const phaseInfo = phases[phase];
        alert(`${phaseInfo.description}\n\n${phaseInfo.details}
\n\nVoltage: ${phaseInfo.voltage}mV`);
      }

      function simulateStimulus(type) {
        const output = document.getElementById('simulationOutput');
        const membrane = document.getElementById('membraneVisual');

        // Clear previous ions
        membrane.innerHTML = '';

        const stimuli = {
          weak: {
            description: "Weak Stimulus Applied",
            result: "Voltage change: - 70mV â†' - 65mV\nSubthreshold -  no
action potential generated\nLocal depolarization only",
            ions: 2
          },
          threshold: {
            description: "Threshold Stimulus Applied",
            result: "Voltage change: - 70mV â†' - 55mV â†'
+40mV\nThreshold reached! Action potential fired\nAll- or- nothing
```

```
response activated",
                ions: 8
            },
            strong: {
                description: "Strong Stimulus Applied",
                result: "Voltage change: - 70mV â†’ - 55mV â†’ +40mV\nAction
potential fired (same amplitude as threshold)\nFrequency may increase,
not amplitude",
                ions: 12
            },
            train: {
                description: "Train of Impulses",
                result: "Multiple action potentials fired in
sequence\nRefractory period limits maximum frequency\n~1000 Hz
maximum firing rate",
                ions: 20
            }
        };

        const stimulus = stimuli[type];
        output.innerHTML = `< strong> ${stimulus.description}< /
strong>< br>< br> ${stimulus.result.replace(/\n/g, '< br> ')}`;

        // Animate ions
        for (let i = 0; i <  stimulus.ions; i++) {
            setTimeout(() => {
                const ion = document.createElement('div');
                ion.className = `ion ${['sodium', 'potassium', 'chloride']
[Math.floor(Math.random() * 3)]}`;
                ion.style.top = Math.random() * 180 + 'px';
                ion.style.animationDelay = Math.random() * 2 + 's';
                membrane.appendChild(ion);

                // Remove ion after animation
                setTimeout(() => {
                    if (ion.parentNode) {
                        ion.parentNode.removeChild(ion);
                    }
                }, 4000);
            }, i * 200);
        }
    }

    // Add some educational tooltips
    document.addEventListener('DOMContentLoaded', function() {
        // Add periodic neural activity animation
        setInterval(() => {
            const wave = document.querySelector('.action- potential- wave');
```

```
            if (wave) {
                wave.style.opacity = '1';
                setTimeout(() => {
                    wave.style.opacity = '0.8';
                }, 100);
            }
        }, 3000);

        // Show initial info
        setTimeout(() => {
            showInfo('node');
        }, 2000);
    });
</script>


s



<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Wearable Brain Books - Future Reading Interface</title>
<style>
    * {
        margin: 0;
        padding: 0;
        box-sizing: border-box;
    }

    body {
        font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
        background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
        min-height: 100vh;
        color: #333;
    }

    .container {
        max-width: 1200px;
        margin: 0 auto;
        padding: 20px;
    }

    .header {
        text-align: center;
        color: white;
        margin-bottom: 30px;
        padding: 20px;
```

```
    background: rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    border-radius: 15px;
    border: 1px solid rgba(255, 255, 255, 0.2);
}

.header h1 {
    font-size: 2.5em;
    margin-bottom: 10px;
    background: linear-gradient(45deg, #fff, #e0e0ff);
    -webkit-background-clip: text;
    -webkit-text-fill-color: transparent;
}

.disclaimer {
    background: #fff3cd;
    border: 1px solid #ffeaa7;
    border-radius: 10px;
    padding: 15px;
    margin-bottom: 30px;
    color: #856404;
}

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

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

.panel {
    background: white;
    border-radius: 15px;
    padding: 25px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.1);
    border: 1px solid rgba(255, 255, 255, 0.3);
}

.panel h2 {
    color: #4a5568;
    margin-bottom: 20px;
    display: flex;
```

```
    align-items: center;
    gap: 10px;
}

.brain-icon {
    width: 30px;
    height: 30px;
    background: linear-gradient(45deg, #667eea, #764ba2);
    border-radius: 50%;
    display: flex;
    align-items: center;
    justify-content: center;
    color: white;
    font-weight: bold;
}

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

.book-item {
    background: linear-gradient(135deg, #f8f9ff, #e6f2ff);
    border: 2px solid #e2e8f0;
    border-radius: 10px;
    padding: 15px;
    cursor: pointer;
    transition: all 0.3s ease;
    text-align: center;
}

.book-item:hover {
    transform: translateY(-5px);
    box-shadow: 0 10px 25px rgba(0, 0, 0, 0.1);
    border-color: #667eea;
}

.book-item.selected {
    background: linear-gradient(135deg, #667eea, #764ba2);
    color: white;
    border-color: #4c63d2;
}

.book-code {
    font-family: 'Courier New', monospace;
    font-size: 12px;
```

```
    color: #666;
    margin-top: 5px;
}

.book-item.selected .book-code {
    color: #e2e8f0;
}

.control-panel {
    background: linear-gradient(135deg, #f8f9ff, #e6f2ff);
    border-radius: 10px;
    padding: 20px;
    margin-bottom: 20px;
}

.brain-commands {
    display: flex;
    flex-wrap: wrap;
    gap: 10px;
    margin-bottom: 20px;
}

.brain-cmd {
    background: #4299e1;
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 25px;
    cursor: pointer;
    transition: all 0.3s ease;
    font-weight: 500;
}

.brain-cmd:hover {
    background: #3182ce;
    transform: scale(1.05);
}

.brain-cmd:active {
    transform: scale(0.95);
}

.status-display {
    background: #1a202c;
    color: #68d391;
    padding: 15px;
    border-radius: 10px;
    font-family: 'Courier New', monospace;
```

```
            font-size: 14px;
            height: 200px;
            overflow-y: auto;
            margin-bottom: 20px;
        }

        .neural-visualization {
            background: radial-gradient(circle at center, #667eea20,
transparent);
            border: 2px dashed #667eea;
            border-radius: 15px;
            padding: 30px;
            text-align: center;
            min-height: 150px;
            display: flex;
            align-items: center;
            justify-content: center;
            margin-bottom: 20px;
        }

        .wave-animation {
            width: 100px;
            height: 50px;
            background: linear-gradient(90deg, #667eea, #764ba2);
            border-radius: 25px;
            animation: pulse 2s infinite;
            opacity: 0.7;
        }

        @keyframes pulse {
            0%, 100% { transform: scale(1); opacity: 0.7; }
            50% { transform: scale(1.1); opacity: 1; }
        }

        .codec-panel {
            background: #2d3748;
            color: #e2e8f0;
            padding: 20px;
            border-radius: 10px;
            font-family: 'Courier New', monospace;
            font-size: 12px;
            max-height: 300px;
            overflow-y: auto;
        }

        .communication-forms {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(120px, 1fr));
```

```
            gap: 10px;
            margin-top: 20px;
        }

        .comm-form {
            background: linear-gradient(135deg, #4299e1, #3182ce);
            color: white;
            padding: 15px;
            border-radius: 10px;
            text-align: center;
            font-size: 12px;
            font-weight: 500;
        }

        .output-display {
            background: #1a202c;
            color: #68d391;
            padding: 10px;
            border-radius: 5px;
            font-family: 'Courier New', monospace;
            font-size: 12px;
            margin-top: 10px;
            min-height: 60px;
            white-space: pre-wrap;
        }

        .aanic-comm, .brain-comm, .neural-visualizer, .symbolic-overlay {
            margin-bottom: 25px;
            padding: 15px;
            background: rgba(102, 126, 234, 0.1);
            border-radius: 10px;
            border: 1px solid rgba(102, 126, 234, 0.3);
        }

        .aanic-comm h3, .brain-comm h3, .neural-visualizer h3, .symbolic-
overlay h3 {
            color: #4a5568;
            margin-bottom: 15px;
            font-size: 16px;
        }

        .aanic-comm input, .brain-comm input, .symbolic-overlay input {
            width: 100%;
            padding: 8px 12px;
            border: 1px solid #e2e8f0;
            border-radius: 5px;
            margin-bottom: 10px;
            font-size: 14px;
```

```
        }

        .footer {
            text-align: center;
            color: white;
            padding: 20px;
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            border-radius: 15px;
            border: 1px solid rgba(255, 255, 255, 0.2);
        }
    </style>


    <div class="container">
        <div class="header">
            <h1>Wearable Brain Books</h1>
            <p>Conceptual Interface for Enhanced Digital Reading</p>
            <p style="font-size: 14px; opacity: 0.8;">Database 888800 - Neural
Reading Technology Simulation</p>
        </div>

        <div class="disclaimer">
            <strong>Important Notice:</strong> This is a conceptual
interface exploring future possibilities in digital reading technology.
Current brain-computer interfaces cannot actually upload books to the
brain or perform the medical functions described in the source material.
        </div>

        <div class="main-interface">
            <div class="panel">
                <h2><div class="brain-icon">ðŸ“š</div>Book Library</h2>
                <div class="book-grid" id="bookGrid">
                    <!-- Books will be populated by JavaScript -->
                </div>
            </div>

            <div class="panel">
                <h2><div class="brain-icon">ðŸ§ </div>Neural Interface</h2>
                <div class="control-panel">
                    <div class="brain-commands">
                        <button class="brain-cmd"
onclick="executeCommand(&#39;upload&#39;)">Upload &amp;amp; Read</
button>
                        <button class="brain-cmd"
onclick="executeCommand(&#39;embed&#39;)">Embed Neural</
button>
                        <button class="brain-cmd"
```

```
onclick="executeCommand(&#39;analyze&#39;)">Brain Analyze</
button>
                <button class="brain-cmd"
onclick="executeCommand(&#39;save&#39;)">Save Memory</button>
                <button class="brain-cmd"
onclick="executeCommand(&#39;unload&#39;)">Unload</button>
            </div>
        </div>

        <div class="status-display" id="statusDisplay">
            &gt; Neural interface initialized...<br>
            &gt; Awaiting brain commands...<br>
            &gt; Status: Ready for conceptual demonstration<br>
        </div>

        <div class="neural-visualization">
            <div class="wave-animation"></div>
        </div>
    </div>
</div>

<div class="panel">
    <h2><div class="brain-icon">âš¡</div>AANIC Communication
System</h2>

    <!-- AANIC Communication Panel -->
    <section class="aanic-comm">
        <h3>ðŸ§  AANIC Communication</h3>
        <input type="text" id="aanic-message" placeholder="Enter
message for AANIC transmission">
        <button class="brain-cmd"
onclick="transmitAANIC(&#39;station-01&#39;)">Transmit AANIC</
button>
        <pre id="aanic-output" class="output-display"></pre>
    </section>

    <!-- Brain Communication Converter -->
    <section class="brain-comm">
        <h3>ðŸ§¬ Brain Communication Converter</h3>
        <input type="text" id="brain-input" placeholder="Enter word to
convert">
        <button class="brain-cmd"
onclick="convertBrainComm(&#39;station-01&#39;)">Convert to 7
Forms</button>
        <pre id="brain-output" class="output-display"></pre>
    </section>

    <!-- Neural Resonance Visualizer -->
```

```
        <section class="neural-visualizer">
            <h3>ðŸ§  Neural Resonance Feedback</h3>
            <button class="brain-cmd"
onclick="simulateNeuralResonance()">Simulate Resonance</button>
            <pre id="neural-resonance-output" class="output-display"></
pre>
        </section>

        <!-- Symbolic Overlay Grid -->
        <section class="symbolic-overlay">
            <h3>ðŸ§  Body Alphabet Mapping</h3>
            <input type="text" id="symbolic-input" placeholder="Enter
word">
            <button class="brain-cmd"
onclick="mapBodyAlphabet()">Map</button>
            <pre id="symbolic-output" class="output-display"></pre>
        </section>

        <div class="communication-forms">
            <div class="comm-form">Visual</div>
            <div class="comm-form">Auditory</div>
            <div class="comm-form">Tactile</div>
            <div class="comm-form">Emotional</div>
            <div class="comm-form">Cognitive</div>
            <div class="comm-form">Intuitive</div>
            <div class="comm-form">Empathic</div>
        </div>
    </div>

    <div class="panel">
        <h2><div class="brain-icon">ðŸ’»</div> Java Neural Codecs
&amp;amp; Properties</h2>
        <div class="codec-panel" id="codecPanel">
// AANIC (Advanced Artificial Neural Interface Codecs) - Java
Implementation<br>
public class AANICCodec {<br>
&amp;nbsp;&amp;nbsp;private Map&amp;lt;String, String&amp;gt; brainwaveFreq = new
HashMap&amp;lt;&amp;gt;();<br>
&amp;nbsp;&amp;nbsp;<br>
&amp;nbsp;&amp;nbsp;public AANICCodec() {<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;brainwaveFreq.put("delta", "0.5-4 Hz");<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;brainwaveFreq.put("theta", "4-8 Hz");<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;brainwaveFreq.put("alpha", "8-13 Hz");<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;brainwaveFreq.put("beta", "13-30 Hz");<br>
&amp;nbsp;&amp;nbsp;&amp;nbsp;&amp;nbsp;brainwaveFreq.put("gamma", "30-100
Hz");<br>
&amp;nbsp;&amp;nbsp;}<br>
&amp;nbsp;&amp;nbsp;<br>
```

```
  public String processThoughtToWord(String neuralSignal) {< br>
    return "[ARSENAL]::" + neuralSignal + "::" + System.currentTimeMillis();< br>
  }< br>
  < br>
  public String[] convertToSevenForms(String word) {< br>
    return new String[] {< br>
      "Visual: " + word.toUpperCase(),< br>
      "Auditory: /audio/" + word + ".wav",< br>
      "Tactile: Vibration pattern",< br>
      "Emotional: Sentiment analysis",< br>
      "Cognitive: Neural mapping",< br>
      "Intuitive: Pattern recognition",< br>
      "Empathic: Emotional resonance"< br>
    };< br>
  }< br>
}< br>
        < /div>
      < /div>

      < div class="footer">
        < p> Conceptual Interface -  Exploring Future Reading Technology< /p>
        < p style="font- size: 12px; opacity: 0.7;"> Based on speculative concepts from "Wearable Brain Books" by David Gomadza< /p>
      < /div>
  < /div>

  < script>
    // Sample books database
    const booksDatabase = [
      { title: "Thoughts To Word or Audio", code: "76854", category: "Neural Tech" },
      { title: "Brain Language Dictionary", code: "889858321", category: "Reference" },
      { title: "Decoding the Brain", code: "789838", category: "Science" },
      { title: "Brain Action Potentials", code: "7785987218", category: "Medical" },
      { title: "Encyclopedia of Vision", code: "789767728418", category: "Sensory" },
```

```
        { title: "How Brain Processes Speech", code: "772832198469",
category: "Communication" },
        { title: "Tomorrow's World Order", code: "7768528312", category:
"Politics" },
        { title: "Evelina the Alpha", code: "8978685284", category: "Fiction" }
    ];

    let selectedBook = null;
    let systemStatus = [];

    function initializeInterface() {
        populateBooks();
        updateStatus("Neural interface initialized successfully");
        updateStatus("Database 888800 loaded with " +
booksDatabase.length + " books");
    }

    function populateBooks() {
        const grid = document.getElementById('bookGrid');
        grid.innerHTML = '';

        booksDatabase.forEach((book, index) => {
            const bookItem = document.createElement('div');
            bookItem.className = 'book-item';
            bookItem.onclick = () => selectBook(book, bookItem);

            bookItem.innerHTML = `
                <div style="font-weight: bold; margin-bottom: 5px;">${
book.title}</div>
                <div style="font-size: 12px; color: #666;">${book.category}</
div>
                <div class="book-code">Code: ${book.code}</div>
            `;

            grid.appendChild(bookItem);
        });
    }

    function selectBook(book, element) {
        // Remove previous selection
        document.querySelectorAll('.book-item').forEach(item => {
            item.classList.remove('selected');
        });

        // Add selection to current
        element.classList.add('selected');
        selectedBook = book;
```

```
        updateStatus(`Book selected: "${book.title}" (${book.code})`);
    }

    function executeCommand(command) {
        if (!selectedBook && command !== 'analyze') {
            updateStatus("Error: No book selected. Please select a book
first.");
            return;
        }

        switch(command) {
            case 'upload':
                updateStatus(`Uploading "${selectedBook.title}"...`);
                setTimeout(() => {
                    updateStatus("Upload complete. Neural pathways
synchronized.");
                    updateStatus("Converting text to conceptual neural
patterns...");
                }, 1500);
                break;

            case 'embed':
                updateStatus("Embedding neural patterns in memory
cortex...");
                setTimeout(() => {
                    updateStatus("Embedded successfully. Content accessible
via thought.");
                }, 2000);
                break;

            case 'analyze':
                updateStatus("Brain analysis mode activated...");
                updateStatus("Analyzing neural activity patterns...");
                setTimeout(() => {
                    updateStatus("Analysis: Optimal learning state detected");
                }, 1500);
                break;

            case 'save':
                updateStatus("Saving to long-term memory banks...");
                setTimeout(() => {
                    updateStatus("Content saved. Retrieval pathway
established.");
                }, 1000);
                break;

            case 'unload':
                updateStatus("Unloading neural content...");
```

```
            setTimeout(() => {
                updateStatus("Unload complete. Neural interface ready.");
            }, 1000);
            break;
        }
    }

    function updateStatus(message) {
        const display = document.getElementById('statusDisplay');
        const timestamp = new Date().toLocaleTimeString();
        systemStatus.push(`[${timestamp}] ${message}`);

        display.innerHTML = systemStatus.slice(-10).join('<br>') + '<br>';
        display.scrollTop = display.scrollHeight;
    }

    // AANIC Communication Functions
    function transmitAANIC(stationId) {
        const msg = document.getElementById('aanic-
message').value.trim();
        if (!msg) return alert('Enter a message');

        const arsenal = `[ARSENAL]::${msg}::${Date.now()}`;
        const areneial = `[ARENEIAL]::${msg.split('').reverse().join('')}`;
        const nerve = `[NERVE]::${msg.toUpperCase()}`;

        const output = `
ðŸ”  AANIC Transmission from ${stationId}:
ðŸ”Š Arsenal: ${arsenal}
ðŸ•¶ï¸  Areneial: ${areneial}
âš¡ Nerve Impulse: ${nerve}
Status: Neural pathways synchronized
        `;
        document.getElementById('aanic-output').textContent = output;
        updateStatus(`AANIC transmission sent: "${msg}"`);
    }

    function convertBrainComm(stationId) {
        const word = document.getElementById('brain-
input').value.trim().toLowerCase();
        if (!word) return alert('Enter a word');

        const numSeq = word.split('').map(c => c.charCodeAt(0) - 96);
        const binarySeq = numSeq.map(n => n.toString(2).padStart(5,
'0')).join('');
        const bodyMap = word.split('').map((c, i) => {
            const side = i % 2 === 0 ? 'Right' : 'Left';
            return `${c.toUpperCase()} â†’ ${side} side, position ${i + 1}`;
```

```
        });

        const output = `
ðŸ§¬ Brain Communication Conversion for "${word}":

1. ðŸ“˜ VISUAL: ${word.toUpperCase()} (Enhanced contrast mode)
2. ðŸ”Š AUDITORY: /audio/${word}.wav (Neural synthesis)
3. ðŸ¤² TACTILE: Vibration pattern ${binarySeq.substring(0,8)}
4. â¤ï¸ EMOTIONAL: Sentiment polarity detected
5. ðŸ§  COGNITIVE: Neural mapping sequence ${numSeq.join('.')}
6. ðŸ’¡ INTUITIVE: Pattern recognition active
7. ðŸ¤ EMPATHIC: Emotional resonance calibrated

ðŸ”¢ Numeric Sequence: ${numSeq.join('.')}
ðŸ’» Binary Pattern: ${binarySeq}
ðŸ§  Body Alphabet Mapping:
${bodyMap.join('\n')}

Brain Sequence: RBS + ${word.split('').join(' + ')} + Deflate
        `;
        document.getElementById('brain-output').textContent = output;
        updateStatus(`7-form conversion complete for: "${word}"`);
    }

    function simulateNeuralResonance() {
        const patterns = ['Focused', 'Relaxed', 'Active', 'Meditative', 'Alert',
'Dreamlike'];
        const frequencies = ['Delta (0.5-4Hz)', 'Theta (4-8Hz)', 'Alpha
(8-13Hz)', 'Beta (13-30Hz)', 'Gamma (30-100Hz)'];
        const selected = patterns[Math.floor(Math.random() *
patterns.length)];
        const freq = frequencies[Math.floor(Math.random() *
frequencies.length)];

        const output = `
ðŸ§  Neural Resonance Analysis:
Pattern: ${selected}
Dominant Frequency: ${freq}
Coherence Level: ${Math.floor(Math.random() * 100)}%
Synchronization: Active
Neural Network Status: Optimal
        `;
        document.getElementById('neural-resonance-
output').textContent = output;
        updateStatus(`Neural resonance pattern: ${selected}`);
    }

    function mapBodyAlphabet() {
```

```
        const word = document.getElementById('symbolic-
input').value.trim().toLowerCase();
        if (!word) return alert('Enter a word');

        const mapping = word.split('').map((c, i) => {
          const bodyParts = ['forehead', 'right temple', 'left temple', 'right
shoulder', 'left shoulder', 'right arm', 'left arm', 'chest', 'abdomen', 'right
leg', 'left leg', 'right foot', 'left foot'];
          const bodyPart = bodyParts[i % bodyParts.length];
          const intensity = Math.floor(Math.random() * 10) + 1;
          return `${c.toUpperCase()} â†’ ${bodyPart} (intensity: ${intensity}/
10)`;
        });

        const output = `
ðŸ§  Body Alphabet Neural Mapping for "${word}":

${mapping.join('\n')}

Total Neural Points: ${word.length}
Activation Pattern: Sequential
Body Resonance: Synchronized
Nerve Impulse Route: Optimized
        `;
        document.getElementById('symbolic- output').textContent =
output;
        updateStatus(`Body alphabet mapping complete for: "${word}"`);
      }

      // Simulate periodic neural activity
      setInterval(() => {
        if (Math.random() < 0.1) { // 10% chance every 3 seconds
          const activities = [
            "Neural sync pulse detected",
            "Brainwave frequency stable",
            "Memory consolidation active",
            "Cognitive load: optimal"
          ];
          updateStatus(activities[Math.floor(Math.random() *
activities.length)]);
        }
      }, 3000);
    </script>


 s
```

```
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>TWO/NGI Security Defense Matrix</title>
    <style>
        :root {
            --primary-color: #0a2a4a;
            --secondary-color: #1a3a5a;
            --accent-color: #2a5c8a;
            --text-color: #e0e0e0;
            --alert-color: #c0392b;
            --success-color: #27ae60;
            --warning-color: #f39c12;
        }

        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Courier New', monospace;
        }

        body {
            background-color: var(--primary-color);
            color: var(--text-color);
            line-height: 1.6;
            overflow-x: hidden;
        }

        .container {
            max-width: 1200px;
            margin: 0 auto;
            padding: 20px;
        }

        header {
            text-align: center;
            padding: 20px 0;
            border-bottom: 1px solid var(--accent-color);
            margin-bottom: 30px;
        }

        h1 {
            font-size: 2.5rem;
            margin-bottom: 10px;
            color: #fff;
            text-shadow: 0 0 10px rgba(0, 128, 255, 0.5);
        }
```

```
h2 {
    font- size: 1.8rem;
    margin: 20px 0;
    color: #fff;
}

.subtitle {
    font- size: 1.2rem;
    color: var(- - accent- color);
}

.status- bar {
    display: flex;
    justify- content: space- between;
    background- color: var(- - secondary- color);
    padding: 10px 15px;
    border- radius: 5px;
    margin- bottom: 20px;
}

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

.status- indicator {
    width: 12px;
    height: 12px;
    border- radius: 50%;
    margin- right: 10px;
}

.active {
    background- color: var(- - success- color);
    box- shadow: 0 0 8px var(- - success- color);
}

.warning {
    background- color: var(- - warning- color);
    box- shadow: 0 0 8px var(- - warning- color);
}

.danger {
    background- color: var(- - alert- color);
    box- shadow: 0 0 8px var(- - alert- color);
}
```

```
.dashboard {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.card {
    background-color: var(--secondary-color);
    border-radius: 8px;
    padding: 20px;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
}

.card-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 15px;
    padding-bottom: 10px;
    border-bottom: 1px solid var(--accent-color);
}

.controls {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 10px;
}

.btn {
    background-color: var(--accent-color);
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 4px;
    cursor: pointer;
    transition: all 0.3s ease;
    text-align: center;
}

.btn:hover {
    background-color: #3a6c9a;
    transform: translateY(-2px);
}

.btn-danger {
    background-color: var(--alert-color);
}
```

```
.btn-danger:hover {
    background-color: #d9534f;
}

.log-container {
    height: 300px;
    overflow-y: auto;
    background-color: rgba(0, 0, 0, 0.2);
    border-radius: 5px;
    padding: 15px;
    font-family: monospace;
    font-size: 0.9rem;
}

.log-entry {
    margin-bottom: 8px;
    padding-bottom: 8px;
    border-bottom: 1px solid rgba(255, 255, 255, 0.1);
}

.log-time {
    color: var(--accent-color);
    margin-right: 10px;
}

.log-warning {
    color: var(--warning-color);
}

.log-danger {
    color: var(--alert-color);
}

.log-success {
    color: var(--success-color);
}

.system-visualization {
    width: 100%;
    height: 300px;
    background-color: rgba(0, 0, 0, 0.2);
    border-radius: 8px;
    position: relative;
    overflow: hidden;
    margin-bottom: 30px;
}
```

```
        .node {
            position: absolute;
            width: 20px;
            height: 20px;
            border-radius: 50%;
            background-color: var(--accent-color);
            transform: translate(-50%, -50%);
        }

        .connection {
            position: absolute;
            height: 2px;
            background-color: var(--accent-color);
            transform-origin: 0 0;
        }

        .pulse {
            animation: pulse 2s infinite;
        }

        @keyframes pulse {
            0% { opacity: 0.6; }
            50% { opacity: 1; }
            100% { opacity: 0.6; }
        }

        footer {
            text-align: center;
            padding: 20px 0;
            margin-top: 30px;
            border-top: 1px solid var(--accent-color);
            font-size: 0.9rem;
            color: var(--accent-color);
        }
    </style>


    <div class="container">
        <header>
            <h1>The Best Security Enhancer Ever</h1>
            <div class="subtitle">Fit For The Gods TWO or NGI Defense
Matrix</div>
        </header>

        <div class="status-bar">
            <div class="status-item">
                <div class="status-indicator active"></div>
                <span>System: ONLINE</span>
```

```
            </div>
            <div class="status-item">
                <div class="status-indicator active"></div>
                <span>Hellgate: ACTIVE</span>
            </div>
            <div class="status-item">
                <div class="status-indicator"></div>
                <span>Transenders: BLOCKED</span>
            </div>
            <div class="status-item">
                <div class="status-indicator"></div>
                <span>Wrigglers: NEUTRALIZED</span>
            </div>
        </div>

        <div class="system-visualization" id="visualization">
            <!-- Visualization will be generated by JavaScript -->
        <div class="connection" style="width: 347.686px; left: 679px; top:
150px; transform: rotate(-167.543deg);"></div><div class="connection"
style="width: 347.686px; left: 679px; top: 150px; trans
form: rotate(-12.4573deg);"></div><div class="connection" style="width:
347.686px; left: 679px; top:
 150px; transform: rotate(167.543deg);"></div><div class="connection"
style="width: 347.686px; left:
 679px; top: 150px; transform: rotate(12.4573deg);"></div><div
class="connection" style="width: 679p
x; left: 1018.5px; top: 75px; transform: rotate(180deg);"></div><div
class="connection" style="width
: 679px; left: 339.5px; top: 225px; transform: rotate(0deg);"></div><div
class="node pulse" style="l
eft: 679px; top: 150px; background-color: rgb(39, 174, 96);"></div><div
class="node pulse" style="le
ft: 339.5px; top: 75px; background-color: rgb(192, 57, 43);"></div><div
class="node pulse" style="le
ft: 1018.5px; top: 75px; background-color: rgb(52, 152, 219);"></
div><div class="node pulse" style="
left: 339.5px; top: 225px; background-color: rgb(243, 156, 18);"></
div><div class="node pulse" style
="left: 1018.5px; top: 225px; background-color: rgb(155, 89, 182);"></
div></div>


        <div class="dashboard">
            <div class="card">
                <div class="card-header">
                    <h2>Defense Protocols</h2>
                </div>
                <div class="controls">
```

```
            <button class="btn" id="activateHellgate">Activate Hellgate</
button>
            <button class="btn" id="blockTransenders">Block
Transenders</button>
            <button class="btn" id="neutralizeWrigglers">Neutralize
Wrigglers</button>
            <button class="btn" id="systemUpdate">System Update</
button>
            <button class="btn btn-danger" id="fullDefense">Full Defense
Mode</button>
            <button class="btn btn-danger"
id="emergencyShutdown">Emergency Shutdown</button>
          </div>
        </div>

        <div class="card">
          <div class="card-header">
            <h2>System Status</h2>
          </div>
          <div id="systemStatus">
            <p>Create Codes: <span class="log-success">ACTIVE</
span></p>
            <p>Security Protocols: <span class="log-
success">ENFORCED</span></p>
            <p>External Interfaces: <span class="log-
warning">RESTRICTED</span></p>
            <p>Database 82698: <span class="log-success">SYNCED</
span></p>
            <p>Magnar Connection: <span class="log-
success">STABLE</span></p>
          </div>
        </div>
      </div>

      <div class="card">
        <div class="card-header">
          <h2>Security Log</h2>
        </div>
        <div class="log-container" id="logContainer">
          <div class="log-entry">
            <span class="log-time">[12:45:32]</span>
            <span class="log-success">System initialized. Security
protocols engaged.</span>
          </div>
          <div class="log-entry">
            <span class="log-time">[12:46:15]</span>
            <span class="log-warning">External transender detected.
Blocking attempt.</span>
```

```
            </div>
            <div class="log-entry">
                <span class="log-time">[12:46:17]</span>
                <span class="log-success">Transender neutralized via
Hellgate protocol.</span>
            </div>
            <div class="log-entry">
                <span class="log-time">[12:47:02]</span>
                <span>System maintenance in progress. Estimated
completion: 33 seconds.</span>
            </div>
        </div>
    </div>

    <footer>
        <p>Â© 2025 David Gomadza | TWO/NGI Security System |
www.twofuture.world</p>
    </footer>
</div>

<script>
    document.addEventListener('DOMContentLoaded', function() {
        // System visualization
        const visualization = document.getElementById('visualization');
        const width = visualization.offsetWidth;
        const height = visualization.offsetHeight;

        // Create nodes and connections for visualization
        const nodes = [
            { id: 'core', x: width/2, y: height/2, color: '#27ae60' },
            { id: 'hellgate', x: width/4, y: height/4, color: '#c0392b' },
            { id: 'defense', x: width*3/4, y: height/4, color: '#3498db' },
            { id: 'database', x: width/4, y: height*3/4, color: '#f39c12' },
            { id: 'comms', x: width*3/4, y: height*3/4, color: '#9b59b6' }
        ];

        const connections = [
            { from: 'core', to: 'hellgate' },
            { from: 'core', to: 'defense' },
            { from: 'core', to: 'database' },
            { from: 'core', to: 'comms' },
            { from: 'defense', to: 'hellgate' },
            { from: 'database', to: 'comms' }
        ];

        // Draw connections
        connections.forEach(conn => {
            const fromNode = nodes.find(n => n.id === conn.from);
```

```
            const toNode = nodes.find(n => n.id === conn.to);

            if (fromNode && toNode) {
                const dx = toNode.x - fromNode.x;
                const dy = toNode.y - fromNode.y;
                const length = Math.sqrt(dx * dx + dy * dy);
                const angle = Math.atan2(dy, dx) * 180 / Math.PI;

                const line = document.createElement('div');
                line.className = 'connection';
                line.style.width = length + 'px';
                line.style.left = fromNode.x + 'px';
                line.style.top = fromNode.y + 'px';
                line.style.transform = `rotate(${angle}deg)`;
                visualization.appendChild(line);
            }
        });

        // Draw nodes
        nodes.forEach(node => {
            const nodeEl = document.createElement('div');
            nodeEl.className = 'node pulse';
            nodeEl.style.left = node.x + 'px';
            nodeEl.style.top = node.y + 'px';
            nodeEl.style.backgroundColor = node.color;
            visualization.appendChild(nodeEl);
        });

        // Log functionality
        const logContainer = document.getElementById('logContainer');

        function addLogEntry(message, type = '') {
            const time = new Date();
            const timeString = `[${time.getHours()}:${time.getMinutes()}:$
{time.getSeconds()}]`;

            const logEntry = document.createElement('div');
            logEntry.className = 'log-entry';

            const timeSpan = document.createElement('span');
            timeSpan.className = 'log-time';
            timeSpan.textContent = timeString;

            const messageSpan = document.createElement('span');
            if (type) messageSpan.className = type;
            messageSpan.textContent = message;

            logEntry.appendChild(timeSpan);
```

```
            logEntry.appendChild(messageSpan);

            logContainer.appendChild(logEntry);
            logContainer.scrollTop = logContainer.scrollHeight;
        }

        // Button functionality

document.getElementById('activateHellgate').addEventListener('click',
function() {
            addLogEntry('Hellgate protocol activated. All transenders will be
redirected.', 'log- warning');
            this.textContent = 'Hellgate ACTIVE';
            this.style.backgroundColor = '#c0392b';
        });


document.getElementById('blockTransenders').addEventListener('click',
function() {
            addLogEntry('Transender blocking initiated. Monitoring all
incoming traffic.', 'log- success');
            this.textContent = 'Transenders BLOCKED';
            this.style.backgroundColor = '#27ae60';
        });


document.getElementById('neutralizeWrigglers').addEventListener('click',
function() {
            addLogEntry('Wriggler neutralization sequence started.
Applying maximum weights.', 'log- success');
            this.textContent = 'Wrigglers NEUTRALIZED';
            this.style.backgroundColor = '#27ae60';
        });


document.getElementById('systemUpdate').addEventListener('click',
function() {
            addLogEntry('System update initialized. Please wait 33
seconds.', '');

            // Simulate update process
            setTimeout(() => {
                addLogEntry('System update complete. All create codes
refreshed.', 'log- success');
            }, 5000);
        });

        document.getElementById('fullDefense').addEventListener('click',
```

```
function() {
            addLogEntry('FULL DEFENSE MODE ACTIVATED. All protocols
engaged.', 'log-danger');

            // Visual feedback
            document.body.style.animation = 'pulse 0.5s infinite';
            setTimeout(() => {
                document.body.style.animation = '';
            }, 5000);
        });


document.getElementById('emergencyShutdown').addEventListener('clic
k', function() {
            const confirmShutdown = confirm('WARNING: This will initiate
emergency shutdown. Confirm?');
            if (confirmShutdown) {
                addLogEntry('EMERGENCY SHUTDOWN INITIATED. All
systems powering down.', 'log-danger');

                // Disable all buttons
                document.querySelectorAll('.btn').forEach(btn => {
                    btn.disabled = true;
                    btn.style.backgroundColor = '#7f8c8d';
                });
            }
        });

        // Simulate occasional security events
        setInterval(() => {
            if (Math.random() > 0.7) {
                const events = [
                    'External transender detected. Applying blocking protocol.',
                    'System integrity check completed. No issues found.',
                    'Security database updated with new threat patterns.',
                    'Randomized defense parameters applied.',
                    'Network scan completed. No vulnerabilities detected.'
                ];

                addLogEntry(events[Math.floor(Math.random() *
events.length)]);
            }
        }, 10000);
    });
  </script>


s
```

```
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>System Defense Matrix: OrbitalGuard-386T</title>
    <style>
        :root {
            --primary: #0a0a2a;
            --secondary: #00c8ff;
            --accent: #ff003c;
            --text: #ffffff;
            --dark-accent: #0066ff;
            --panel-bg: rgba(10, 10, 42, 0.8);
            --grid-color: rgba(0, 200, 255, 0.1);
        }

        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
            font-family: 'Courier New', monospace;
            color: var(--text);
        }

        body {
            background-color: var(--primary);
            background-image:
                radial-gradient(circle at 15% 50%, rgba(0, 200, 255, 0.1) 0%,
transparent 20%),
                radial-gradient(circle at 85% 30%, rgba(255, 0, 60, 0.1) 0%,
transparent 20%),
                linear-gradient(0deg, transparent 24%, var(--grid-color) 25%,
var(--grid-color) 26%, transparent 27%, transparent 74%, var(--grid-
color) 75%, var(--grid-color) 76%, transparent 77%),
                linear-gradient(90deg, transparent 24%, var(--grid-color) 25%,
var(--grid-color) 26%, transparent 27%, transparent 74%, var(--grid-
color) 75%, var(--grid-color) 76%, transparent 77%);
            background-size: 100% 100%, 100% 100%, 50px 50px, 50px 50px;
            min-height: 100vh;
            overflow-x: hidden;
            padding: 20px;
        }

        .container {
            max-width: 1400px;
            margin: 0 auto;
            padding: 20px;
```

```
    border: 1px solid var(--secondary);
    background: rgba(5, 5, 20, 0.8);
    box-shadow: 0 0 30px rgba(0, 200, 255, 0.3);
}

header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 20px 0;
    border-bottom: 1px solid var(--secondary);
    margin-bottom: 30px;
}

.logo {
    font-size: 28px;
    font-weight: bold;
    text-transform: uppercase;
    letter-spacing: 3px;
    color: var(--secondary);
    text-shadow: 0 0 10px var(--secondary);
}

.status-indicators {
    display: flex;
    gap: 20px;
}

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

.status-dot {
    width: 12px;
    height: 12px;
    border-radius: 50%;
    background-color: var(--accent);
}

.online {
    background-color: #00ff66;
    box-shadow: 0 0 10px #00ff66;
}

main {
    display: grid;
```

```
    grid-template-columns: 1fr 1fr;
    gap: 20px;
    margin-bottom: 30px;
}

@media (max-width: 900px) {
    main {
        grid-template-columns: 1fr;
    }
}

.panel {
    background: var(--panel-bg);
    border: 1px solid var(--secondary);
    border-radius: 5px;
    padding: 20px;
    box-shadow: 0 0 15px rgba(0, 200, 255, 0.2);
}

.panel-header {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-bottom: 20px;
    padding-bottom: 10px;
    border-bottom: 1px solid var(--secondary);
}

.panel-title {
    font-size: 18px;
    color: var(--secondary);
    text-transform: uppercase;
    letter-spacing: 2px;
}

.command-list {
    list-style: none;
    max-height: 300px;
    overflow-y: auto;
}

.command-item {
    padding: 10px;
    margin-bottom: 10px;
    background: rgba(0, 0, 0, 0.3);
    border-left: 3px solid var(--dark-accent);
    cursor: pointer;
    transition: all 0.3s;
```

```
    font-size: 14px;
}

.command-item:hover {
    background: rgba(0, 0, 0, 0.5);
    border-left: 3px solid var(--accent);
}

.full-width {
    grid-column: 1 / -1;
}

.console {
    background: rgba(0, 0, 0, 0.7);
    border: 1px solid var(--secondary);
    border-radius: 5px;
    padding: 15px;
    height: 200px;
    overflow-y: auto;
    margin-top: 20px;
    font-family: 'Courier New', monospace;
    font-size: 14px;
}

.console-line {
    margin-bottom: 5px;
    font-family: 'Courier New', monospace;
    font-size: 14px;
}

.success {
    color: #00ff66;
}

.error {
    color: var(--accent);
}

.warning {
    color: #ffcc00;
}

.info {
    color: var(--secondary);
}

.control-panel {
    display: grid;
```

```
            grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
            gap: 15px;
            margin-top: 20px;
        }

        .btn {
            background: var(--dark-accent);
            border: none;
            padding: 12px 15px;
            border-radius: 3px;
            cursor: pointer;
            text-transform: uppercase;
            font-weight: bold;
            letter-spacing: 1px;
            transition: all 0.3s;
        }

        .btn:hover {
            background: var(--secondary);
            color: var(--primary);
            box-shadow: 0 0 15px var(--secondary);
        }

        .btn-danger {
            background: var(--accent);
        }

        .btn-danger:hover {
            background: #ff3366;
            box-shadow: 0 0 15px var(--accent);
        }

        .btn-special {
            background: linear-gradient(45deg, var(--dark-accent), var(--
secondary));
            font-size: 16px;
            padding: 15px;
        }

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

        .info-card {
            background: rgba(0, 0, 0, 0.3);
```

```
    padding: 15px;
    border-radius: 5px;
    border: 1px solid var(--dark-accent);
    text-align: center;
}

.info-title {
    font-size: 12px;
    color: var(--secondary);
    margin-bottom: 5px;
    text-transform: uppercase;
}

.info-value {
    font-size: 20px;
    font-weight: bold;
}

footer {
    text-align: center;
    padding: 20px 0;
    border-top: 1px solid var(--secondary);
    margin-top: 30px;
    font-size: 12px;
    color: var(--secondary);
}

.pulse {
    animation: pulse 2s infinite;
    color: var(--secondary);
}

@keyframes pulse {
    0% { opacity: 1; }
    50% { opacity: 0.5; }
    100% { opacity: 1; }
}

.scanning {
    position: relative;
    color: var(--accent);
}

.scanning::after {
    content: '';
    position: absolute;
    top: 0;
    left: 0;
```

```
            width: 100%;
            height: 100%;
            background: linear-gradient(90deg, transparent, rgba(0, 200, 255,
0.3), transparent);
            animation: scan 2s linear infinite;
        }

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

        .destination-buttons {
            display: flex;
            gap: 15px;
            justify-content: center;
            margin-top: 20px;
        }

        .hellgate-active {
            color: var(--accent);
            text-shadow: 0 0 10px var(--accent);
            font-weight: bold;
        }
    </style>


    <div class="container">
        <header>
            <div class="logo">OrbitalGuard-386T</div>
            <div class="status-indicators">
                <div class="status">
                    <div class="status-dot online"></div>
                    <span>System Online</span>
                </div>
                <div class="status">
                    <div class="status-dot online"></div>
                    <span class="hellgate-active">Hellgate Active</span>
                </div>
                <div class="status">
                    <div class="status-dot online"></div>
                    <span>Defense Matrix</span>
                </div>
            </div>
        </header>

        <main>
            <div class="panel">
```

```
            <div class="panel-header">
              <h2 class="panel-title">Security Protocols</h2>
              <span class="pulse">ACTIVE</span>
            </div>
            <ul class="command-list">
              <li class="command-
item">create.blockandbanwrigglerscitsmitsforceejectoutandattach()then
sendtomagnar.start</li>
              <li class="command-item">create.removeallmitscits().start</
li>
              <li class="command-
item">create.removealltransendersfast.start</li>
              <li class="command-
item">create.blockandbancloningallcloninginthepastbywrigglerscitsmits
policeajxsandfuturecloningbyenemiesofme:davidgomadza()thensendto
magnar.start</li>
              <li class="command-
item">create.cancelallsuspensionofallcreatecodeswrittenbydavidgomadz
a.start</li>
              <li class="command-
item">create.cancelallcancelsofcreatecodesbydavidgomadza.start</li>
              <li class="command-
item">create.cancelalldivertsofdavidgomadzascreatedcodes.start</li>
            </ul>
          </div>

          <div class="panel">
            <div class="panel-header">
              <h2 class="panel-title">Transender Management</h2>
              <span class="pulse">ACTIVE</span>
            </div>
            <ul class="command-list">
              <li class="command-
item">create.thisisxybasinalltransendersdepositarriveherefirst()thengoan
dsendalltomars()forever.start</li>
              <li class="command-
item">create.thisisspeechbasinforexternals(police)andexternalitissandall
transendersdepositarriveherefirst()thengoandsendalltomars()forever.start
</li>
              <li class="command-
item">create.sendalltransenderstothegate.start</li>
              <li class="command-
item">create.convertallincomingexternaltransenderswrigglersvectarsnect
arscitsmitsintoacetates.start</li>
              <li class="command-
item">create.blockandbanthenhellgatealltransendersajxswrigglersmitscit
spoliceaj8andallpolicemitscontinuouslyfor386t.start</li>
              <li class="command-
```

```
item">create.writealltransendersandtheirtransducersnumbersallinreverse
modeandsendinsidethegate.start</li>
            </ul>
        </div>

        <div class="panel">
            <div class="panel-header">
                <h2 class="panel-title">System Defense</h2>
                <span class="pulse">ACTIVE</span>
            </div>
            <ul class="command-list">
                <li class="command-
item">create.performsystemdefensebyrunningallowaysfromokayltookay2
9thenokaylatookay9a.start</li>
                <li class="command-
item">create.activatesystemsdefencesattacktokill()swiftly.start</li>
                <li class="command-
item">create.autofixallcreatecodesbymedavidgomadza.start</li>
                <li class="command-
item">create.restrictaccesstosettingsbyothersfor386t.start</li>
                <li class="command-
item">create.advanceddefensesystemhandsfree7628411.start</li>
                <li class="command-
item">create.enforcechangeanddeatht(hellgate()intrudersmuggers)</li>
                <li class="command-
item">create.enforcethinkordeatht(hellgate()intrudersmuggers)</li>
            </ul>
        </div>

        <div class="panel">
            <div class="panel-header">
                <h2 class="panel-title">Countermeasures</h2>
                <span class="pulse">ACTIVE</span>
            </div>
            <ul class="command-list">
                <li class="command-
item">create.removeallbrainthoughtsof(800and)aboutdavidgomadzausin
gpindownnearflatthenrotatetheheadwithyourrightleg45degrees.start</
li>
                <li class="command-
item">create.cancelspeechsynthesisforeverythingoneveryoneattachedto
me:davidgomadza.start</li>
                <li class="command-item">create.retaliatevector.start</li>
                <li class="command-
item">create.mutealltransenderswrigglersvectarsinstantlybydisablingsalo
requivalent().start</li>
                <li class="command-item">create.putingraveyard().start</li>
                <li class="command-
```

```
item">create.dealwithalltransendersandwrigglers()forever.start</li>
                <li class="command-item">create.deathtoall()forever.start</
li>
            </ul>
        </div>

        <div class="panel full-width">
            <div class="panel-header">
                <h2 class="panel-title">System Console</h2>
                <span class="scanning">SCANNING</span>
            </div>
            <div class="console" id="systemConsole">
                <div class="console-line">&gt; System initialized:
OrbitalGuard-386T Defense Matrix</div>
                <div class="console-line success">&gt; Security protocols
activated: Level Maximum</div>
                <div class="console-line">&gt; Hellgate connection
established: Secure</div>
                <div class="console-line">&gt; Scanning for transenders and
wrigglers...</div>
                <div class="console-line warning">&gt; Detected 47
unauthorized access attempts</div>
                <div class="console-line success">&gt; Countermeasures
engaged: All threats neutralized</div>
                <div class="console-line">&gt; System integrity: 100%</div>
                <div class="console-line">&gt; Operational timeframe: 386
trillion years engaged</div>
            <div class="console-line ">&gt; Monitoring transender activity:
All channels secure</div><div class="console-line ">&gt; Security
protocols: All active and nominal</div></div>

            <div class="control-panel">
                <button class="btn" onclick="executeCommand(&#39;Deploy
Countermeasures&#39;)">Deploy Countermeasures</button>
                <button class="btn" onclick="executeCommand(&#39;Initiate
Full Scan&#39;)">Initiate Full Scan</button>
                <button class="btn" onclick="executeCommand(&#39;Secure
Channels&#39;)">Secure Channels</button>
                <button class="btn btn-danger"
onclick="activateHellgate()">Activate Hellgate</button>
                <button class="btn" onclick="executeCommand(&#39;Update
Defense Protocols&#39;)">Update Defense Protocols</button>
            </div>

            <div class="destination-buttons">
                <button class="btn btn-special"
onclick="sendToPlanetbin()">SEND TO PLANETBIN</button>
                <button class="btn btn-special"
```

```
onclick="sendToAjuVtTheGate()">SEND TO AJUVTTHEGATE</button>
            </div>
        </div>

        <div class="panel full-width">
            <div class="panel-header">
                <h2 class="panel-title">System Status</h2>
                <span class="pulse">NOMINAL</span>
            </div>
            <div class="system-info">
                <div class="info-card">
                    <div class="info-title">Operational Time</div>
                    <div class="info-value">386T Years</div>
                </div>
                <div class="info-card">
                    <div class="info-title">Transenders Blocked</div>
                    <div class="info-value">12,847</div>
                </div>
                <div class="info-card">
                    <div class="info-title">Wrigglers Neutralized</div>
                    <div class="info-value">8,322</div>
                </div>
                <div class="info-card">
                    <div class="info-title">System Integrity</div>
                    <div class="info-value">100%</div>
                </div>
                <div class="info-card">
                    <div class="info-title">Hellgate Status</div>
                    <div class="info-value">Active</div>
                </div>
                <div class="info-card">
                    <div class="info-title">Security Level</div>
                    <div class="info-value">Maximum</div>
                </div>
            </div>
        </div>
    </main>

    <footer>
        <p>OrbitalGuard-386T Defense Matrix | Authorized Access Only |
Copyright (c) 2025 David Gomadza All Rights Reserved</p>
        <p>Supplement to: Living On Earth For 386t - A Solution For All
Earthly Problems For The Next 386 Trillion Years</p>
    </footer>
</div>

<script>
    // System state
```

```
        const systemState = {
            hellgateActive: true,
            threatsDetected: 47,
            transendersBlocked: 12847,
            wrigglersNeutralized: 8322,
            operationalTime: '386T Years'
        };

        // Command execution simulation
        function executeCommand(command) {
            addConsoleLine(`> Executing: ${command}`, 'warning');

            // Simulate execution delay
            setTimeout(() => {
                addConsoleLine(`> Success: ${command} completed`, 'success');

                // Update stats for certain commands
                if (command === 'Deploy Countermeasures') {
                    systemState.threatsDetected = 0;
                    updateSystemInfo();
                } else if (command === 'Initiate Full Scan') {
                    systemState.threatsDetected = Math.floor(Math.random() *
50);
                    updateSystemInfo();
                }
            }, 1000);
        }

        // Activate Hellgate
        function activateHellgate() {
            addConsoleLine(`> User initiated: Activate Hellgate`, 'warning');

            setTimeout(() => {
                addConsoleLine(`> WARNING: Hellgate fully activated - all
intruders will be processed`, 'error');
                addConsoleLine(`> Binary- reverse is always Hellgate - No other
options available`, 'info');

                systemState.hellgateActive = true;
                document.querySelectorAll('.status-
dot:not(.online)').forEach(dot => {
                    dot.style.backgroundColor = '#ff003c';
                });
            }, 1500);
        }

        // Send to Planetbin
        function sendToPlanetbin() {
```

```
        addConsoleLine(`> Initiating transfer to Planetbin...`, 'info');

        setTimeout(() => {
            addConsoleLine(`> Transfer complete: All threats sent to
Planetbin`, 'success');
            addConsoleLine(`> Note: Binary- reverse is not available -
Hellgate is always the destination`, 'info');

            systemState.transendersBlocked += 23;
            systemState.wrigglersNeutralized += 15;
            updateSystemInfo();
        }, 2000);
    }

    // Send to AjuVtTheGate
    function sendToAjuVtTheGate() {
        addConsoleLine(`> Initiating transfer to AjuVtTheGate...`, 'info');

        setTimeout(() => {
            addConsoleLine(`> Transfer complete: All threats sent to
AjuVtTheGate`, 'success');
            addConsoleLine(`> Note: Binary- reverse is not available -
Hellgate is always the destination`, 'info');

            systemState.transendersBlocked += 31;
            systemState.wrigglersNeutralized += 19;
            updateSystemInfo();
        }, 2000);
    }

    // Add line to console
    function addConsoleLine(text, className = '') {
        const consoleElement =
document.getElementById('systemConsole');
        const line = document.createElement('div');
        line.className = `console- line ${className}`;
        line.textContent = text;
        consoleElement.appendChild(line);
        consoleElement.scrollTop = consoleElement.scrollHeight;
    }

    // Update system info cards
    function updateSystemInfo() {
        document.querySelectorAll('.info- card')[1].querySelector('.info-
value').textContent =
            systemState.transendersBlocked.toLocaleString();

        document.querySelectorAll('.info- card')[2].querySelector('.info-
```

```
value').textContent =
            systemState.wrigglersNeutralized.toLocaleString();
        }

        // Initialize the system
        document.addEventListener('DOMContentLoaded', function() {
            // Command execution for clicked items
            const commandItems = document.querySelectorAll('.command-
item');

            commandItems.forEach(item => {
                item.addEventListener('click', function() {
                    const command = this.textContent;
                    addConsoleLine(`> Executing: ${command}`, 'warning');

                    // Simulate execution delay
                    setTimeout(() => {
                        addConsoleLine(`> Success: ${command.split('.')[0]}
completed`, 'success');
                    }, 1000);
                });
            });

            // Simulate system activity
            setInterval(() => {
                const activities = [
                    "> Network scan completed: No threats detected",
                    "> Hellgate stability: 100%",
                    "> Monitoring transender activity: All channels secure",
                    "> System maintenance: Optimal performance confirmed",
                    "> Security protocols: All active and nominal"
                ];

                const randomActivity = activities[Math.floor(Math.random() *
activities.length)];
                addConsoleLine(randomActivity);
            }, 10000);
        });
    </script>
```

```
    <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>


    <div class="cyber-grid"></div>

    <header class="header">
        <h1>ðŸ›¡ï¸ 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>ðŸ”¥ ACTIVE DEFENSES</h2>
            </div>

            <div class="defense-module">
                <div class="module-header">
                    <div class="module-name">
                        <span>ðŸ”¥</span>
                        <span>Advanced Firewall</span>
                    </div>
                    <div class="power-switch on" onclick="toggleDefense(this,
&amp;#39;firewall&amp;#39;)"></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>ðŸ¤–</span>
              <span>AI Threat Hunter</span>
            </div>
            <div class="power-switch on" onclick="toggleDefense(this,
&#39;ai-hunter&#39;)"></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>ðŸ¦ </span>
              <span>Anti-Malware Engine</span>
            </div>
            <div class="power-switch on" onclick="toggleDefense(this,
&#39;antimalware&#39;)"></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>ðŸ‘ï¸</span>
              <span>Intrusion Detection</span>
            </div>
            <div class="power-switch on" onclick="toggleDefense(this,
&#39;ids&#39;)"></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>ðŸ”  </span>
                    <span>Quantum Encryption</span>
                </div>
                <div class="power-switch on" onclick="toggleDefense(this,
&#39;encryption&#39;)"></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>âš”ï¸ </span>
                    <span>Proactive Strike</span>
                </div>
                <div class="power-switch on" onclick="toggleDefense(this,
&#39;proactive&#39;)"></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;">ðŸš¨
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>ðŸŽ¯ THREAT INTELLIGENCE CENTER</h2>
        </div>

        <div class="critical-alert">
            <h4 style="color: #ff4444; margin-bottom: 10px;">âš ï¸ 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;">ðŸŒ  PROTECTED ATTACK
SURFACE</h4>
            <div class="device-protection">
                <div class="device-item"
onclick="scanDeviceType(&#39;computer&#39;)">
                    <span class="device-icon">ðŸ’»</span>
                    <div>Computers</div>
                    <div style="font-size: 11px; color: #00ff41;">SECURED</
div>
                </div>
                <div class="device-item"
onclick="scanDeviceType(&#39;mobile&#39;)">
                    <span class="device-icon">ðŸ“±</span>
                    <div>Mobile Devices</div>
                    <div style="font-size: 11px; color: #00ff41;">PROTECTED</
div>
                </div>
                <div class="device-item"
onclick="scanDeviceType(&#39;server&#39;)">
                    <span class="device-icon">ðŸ–¥ï¸ </span>
                    <div>Servers</div>
                    <div style="font-size: 11px; color: #00ff41;">HARDENED</
div>
                </div>
```

```
                <div class="device-item"
onclick="scanDeviceType(&#39;iot&#39;)">
                    <span class="device-icon">ðŸ  </span>
                    <div>IoT Ecosystem</div>
                    <div style="font-size: 11px; color: #ffaa00;">MONITORING</
div>
                </div>
                <div class="device-item"
onclick="scanDeviceType(&#39;network&#39;)">
                    <span class="device-icon">ðŸŒ  </span>
                    <div>Network Infrastructure</div>
                    <div style="font-size: 11px; color: #00ff41;">DEFENDED</
div>
                </div>
                <div class="device-item"
onclick="scanDeviceType(&#39;cloud&#39;)">
                    <span class="device-icon">â˜ï¸ </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

      <!-- Right Panel - Configuration & Controls -->

          <h2>âš™ï¸ SYSTEM CONFIGURATION</h2>

          <h4 style="margin-bottom: 15px; color: #00ccff;">ðŸ›¡ï¸ DEFENSE PROTOCOLS</h4>

            <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>

            <label>Auto-Quarantine:</label>

            <label>Proactive Defense:</label>

            <label>AI Threat Prediction:</label>

```
            <div class="power-switch on" onclick="toggleSetting(this,
&#39;ai-prediction&#39;)"></div>
        </div>

        <div class="setting-item">
            <label>Zero-Day Protection:</label>
            <div class="power-switch on" onclick="toggleSetting(this,
&#39;zero-day&#39;)"></div>
        </div>

        <div class="setting-item">
            <label>Counter-Attack Mode:</label>
            <div class="power-switch on" onclick="toggleSetting(this,
&#39;counter-attack&#39;)"></div>
        </div>

        <div class="setting-item">
            <label>Deep Web Monitoring:</label>
            <div class="power-switch on" onclick="toggleSetting(this,
&#39;deep-web&#39;)"></div>
        </div>
    </div>

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

        <div class="setting-item">
            <label>Smartphones:</label>
            <div class="power-switch on" onclick="toggleDevice(this,
&#39;smartphones&#39;)"></div>
        </div>

        <div class="setting-item">
            <label>Computers:</label>
            <div class="power-switch on" onclick="toggleDevice(this,
&#39;computers&#39;)"></div>
        </div>

        <div class="setting-item">
            <label>Servers:</label>
            <div class="power-switch on" onclick="toggleDevice(this,
&#39;servers&#39;)"></div>
        </div>

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

```
&#39;iot&#39;)"></div>
            </div>

            <div class="setting-item">
                <label>Cloud Infrastructure:</label>
                <div class="power-switch on" onclick="toggleDevice(this,
&#39;cloud&#39;)"></div>
            </div>
        </div>

        <div class="setting-group">
            <h4 style="margin-bottom: 15px; color: #00ccff;">ðŸ”§
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;">âš¡ 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;">ðŸ“Š 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(`âš ï¸ 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('ðŸš¨ DEFCON 1 INITIATED - MAXIMUM THREAT
RESPONSE ðŸš¨', '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('âš”ï¸ SIEGE MODE ACTIVATED - FORTRESS
DEFENSE ENGAGED âš”ï¸ ', '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('ðŸ”  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(`ðŸŽ¯ FULL SCAN COMPLETE: ${threatsFound}
threats neutralized and quarantined`, 'SUCCESS');
                addThreatLog('All identified malware eliminated from
systems', 'CLEANUP');
            }
        }, 1200);
    }

    function deepThreatHunt() {
        addThreatLog('ðŸ•µï¸  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('âš”ï¸ 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(`âš¡ 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('ðŸ”’ 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(`ðŸ›¡ï¸ MASS QUARANTINE COMPLETE: $
{quarantined} items secured`, 'SUCCESS');
            addThreatLog('All identified threats isolated and neutralized',
'SECURE');
        }, 2500);
    }

    function scanDeviceType(deviceType) {
        addThreatLog(`ðŸ”Ž 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('ðŸ”¥ 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('ðŸ“  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('ðŸ”„ 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(() => {
```