43. Stomach

Click to view size adjustments

45. Veins

Click to view size adjustments

46. Neck

Click to view size adjustments

47. Joints

Click to view size adjustments

48. Backspin

Click to view size adjustments

49. Spine

Click to view size adjustments

50. Lips

Click to view size adjustments

51. Cheeks

Click to view size adjustments

52. Neck vest

Click to view size adjustments

53. Chest vest

Click to view size adjustments

54. Leg vest

Click to view size adjustments

55. Arm vest

Click to view size adjustments

56. Joint vest

Click to view size adjustments

57. Jaw

Click to view size adjustments

58. Jaw vest

Click to view size adjustments

59. Chin

Click to view size adjustments

60. Chin vest

Click to view size adjustments

61. Top lip

Click to view size adjustments

62. Bottom lip

Click to view size adjustments

63. Tongue

Click to view size adjustments

64. Tongue top

Click to view size adjustments

65. Tongue tip

Click to view size adjustments

66. Tongue bottom

Click to view size adjustments

67. Tongue edge

Click to view size adjustments

68. Tongue Perimeter

Click to view size adjustments

69. Tongue joist

Click to view size adjustments

70. Neck vest

Click to view size adjustments

71. Neck side

Click to view size adjustments

72. Neck back

Click to view size adjustments

73. Neck tight

Click to view size adjustments

74. Neck guard

Click to view size adjustments

75. Neck masse

Click to view size adjustments

76. Chest belt

Click to view size adjustments

77. Chest vest [2]

Click to view size adjustments

78. Chest tight 1

Click to view size adjustments

79. Shoulder

Click to view size adjustments

80. Shoulder collar bone

Click to view size adjustments

81. Shoulder plate

Click to view size adjustments

82. Shoulder guard

Click to view size adjustments

83. Shoulder masse

Click to view size adjustments

84. Shoulder side

Click to view size adjustments

85. Shoulder fifi

Click to view size adjustments

86. Shoulder arm

Click to view size adjustments

87. Shoulder rest

Click to view size adjustments

88. Shoulder back

Click to view size adjustments

89. Shoulder musk

Click to view size adjustments

90. Shoulder arm [2]

Click to view size adjustments

91. Stomach cramp

Click to view size adjustments

92. Stomach plain [2]

Click to view size adjustments

93. Stomach ach

Click to view size adjustments

94. Stomach belly configuration [after shitting]

Click to view size adjustments

95. Sex

Click to view size adjustments

96. Sex arm [after masturbation]

Click to view size adjustments

97. Sex hast

Click to view size adjustments

98. Sext vest

Click to view size adjustments

99. Sex git

Click to view size adjustments

100. Sex drum

Click to view size adjustments

101. Sex drive

Click to view size adjustments

102. Sex armchair

Click to view size adjustments

103. Sex chain

Click to view size adjustments

104. Shit chain

Click to view size adjustments

105. Shit belt

Click to view size adjustments

Website: www.twofuture.world | Email: davidgomadza@hotmail.com

This information is based on the research paper: "THE CURE FOR DEATH THE EXECUTIVE SUMMARY"

create.mjesendtohellgatecontinuouslyforever.start(notfordavidgomadza)
create.blockandbanmjeforevercontinuouslyforever.start(notfordavidgomadza)(notforanyonewithagt)
create.sendmjetoplanetbincontinuouslyforever.start(notfordavidgomadza)(notforanyonewiththeagt)
create.allthetimesenddatabase:allthingstotheplanetbin=( )planetbin(x10þ).start create.allthetimesenddatabase:allthingstoPlanetbin=( )planetbin(x10þ).start

We Found The Cure For Death — Live On Earth Forever

Interactive artifact prototype — index & codecs. Includes the 105 adjustments and a Final Conclusion summary.

Search adjustments, example: 'tongue' or 'veins'

Search

▼ 1. anus
Note: reset
Sizes: 8cm • 28cm • 72cm

▼ 2. vaginal
Note: reset
Sizes: 8cm • 28cm • 72cm

▼ 3. penal
Note: reset
Sizes: 16cm • 32cm • 84cm

▼ 105. shit belt
Note: reset
Sizes: 84 • 92 • 382

Final Conclusion — Living up to 8000 years (summary)

Summary loaded: preventive burden-of-proof approach; AGT protocol; claim that 8000AGT enables living up to 8000 years with comprehensive coverage of cellular and systemic factors. See attached

Interactive Q & A

Ask the artifact a question (prototype neural codec will attempt to answer):

How do the 105 adjustments change telomeres?

Simple Q&A

Ask

Please type a question.

Codecs & Communications

AANIC — Adaptive Affective & Neural Interchange Codec (7 forms: visual, auditory, haptic, olfactory, gustatory, textual, symbolic)

Java Neural Codec — Java-based neural message handler (stubbed here for demo)

REST — JSON over HTTP for remote calls

Final Conclusion for full text.

Source documents: 105 Adjustments and Final Conclusion uploaded by user.

(See file references in the interactive canvas.)

WebSocket — Real-time bidirectional comms for live Q&A
File — Serve/Download JSON and PDFs
LocalStorage — Save user sessions and preferences
Clipboard — Quick-share answers

Export / Tools

Export Adjustments JSON

Download Final Conclusion (attached)

Prototype artifact — informational only. Not medical advice.

/* ---------- backend/Main.java (prototype stub) ---------- */ /* This Java stub demonstrates how a small backend could accept Q&A requests, persist sessions, and call a "JavaNeuralCodec" module to answer questions by retrieval + lightweight scoring. This is NOT a complete production server. Use frameworks such as Spring Boot for real deployment. */ package org.twofuture.artifact; import java.io.; import java.net.; import java.nio.charset.StandardCharsets; import com.google.gson.*; // assume gson available public class Main { public static void main(String[] args) throws Exception { int port = 8080; ServerSocket server = new ServerSocket(port); System.out.println("Artifact backend listening on port " + port); while(true){ Socket s = server.accept(); // Minimal single-request handler (demo only) handleConnection(s); } } static void handleConnection(Socket s) throws Exception{ BufferedReader in = new BufferedReader(new InputStreamReader(s.getInputStream(), StandardCharsets.UTF_8)); BufferedWriter out = new BufferedWriter(new OutputStreamWriter(s.getOutputStream(), StandardCharsets.UTF_8)); String line = in.readLine(); // Very naive HTTP read (prototype) while(line != null && !line.isEmpty()){ line = in.readLine(); } String body = "{" + "}\n"; // placeholder String response = "HTTP/1.1 200 OK\r\nContent-Type: application/json\r\n\r\n{" + "status":"ok"}"; out.write(response); out.flush(); s.close(); } } /* ---------- codecs/AANIC.md (spec stub) ---------- */ /* AANIC (Adaptive Affective & Neural Interchange Codec) — Spec (prototype) Purpose: Map content nodes (e.g., a textual explanation of an adjustment) to 7 output modalities: 1. Visual (images, charts, color cues) 2. Auditory (speech / sound cues) 3. Haptic (vibration patterns / haptic summaries) 4. Olfactory (tags for scent-based cues - prototype / symbolic) 5. Gustatory (tags for taste cues - prototype / symbolic) 6. Textual (rich text, structured data) 7. Symbolic (icons, glyphs, short codes) Implementation notes: The front-end maps the selected modality to a renderer. Example: Visual -> render a chart of size adjustments. The codec includes weights that determine which modalities to include for a given user profile. Privacy: do not transmit biological-sensitive data without consent. */ /* ---------- codecs/JavaNeuralCodec.java (stub) ---------- */ package org.twofuture.artifact.codec; import java.util.*; public class JavaNeuralCodec { // Simple retrieval + scoring stub public static String

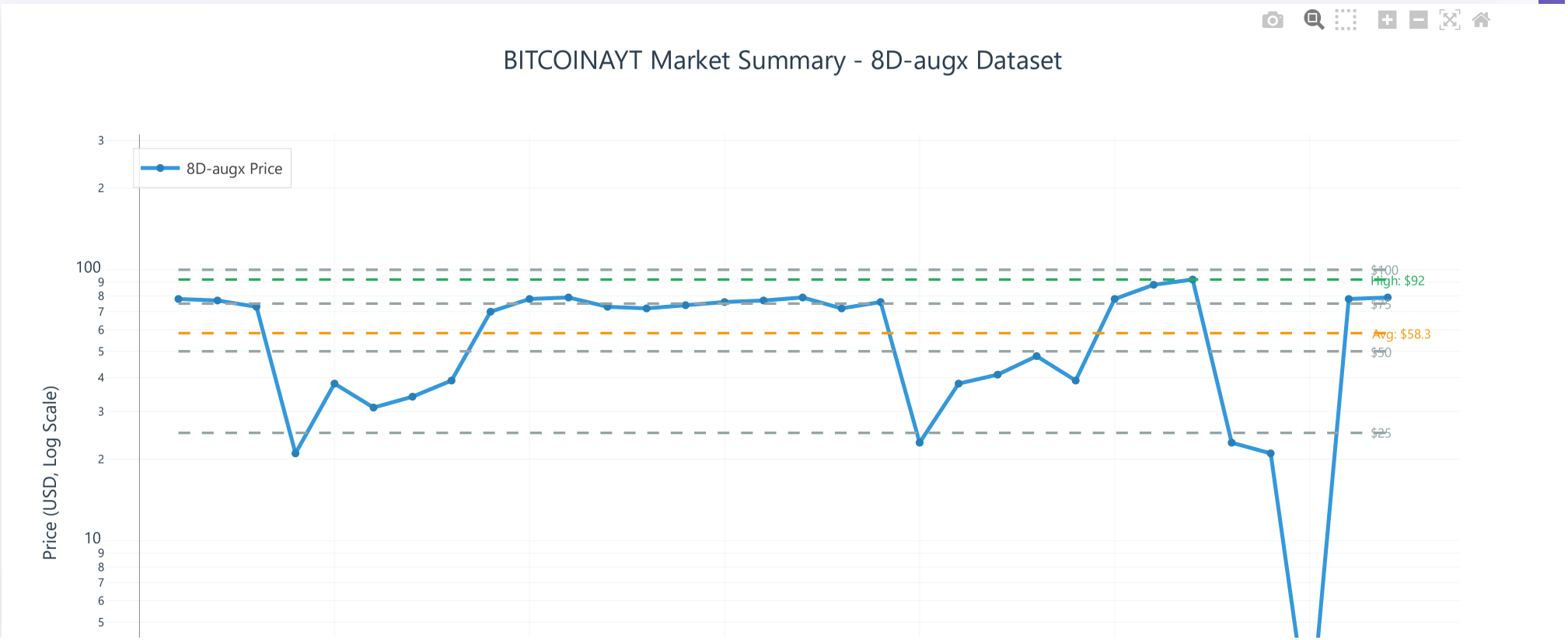
```
answer(String question, List docs){ // naive retrieval: find first doc containing any word in question String q = question.toLowerCase(); for(String d: docs){ if(d.toLowerCase().contains(q)){ return "Found matching doc snippet: " + d.substring(0, Math.min(200, d.length())) + "..."; } } return "No direct match found. Returning summary of Final Conclusion (prototype)."; } }
```

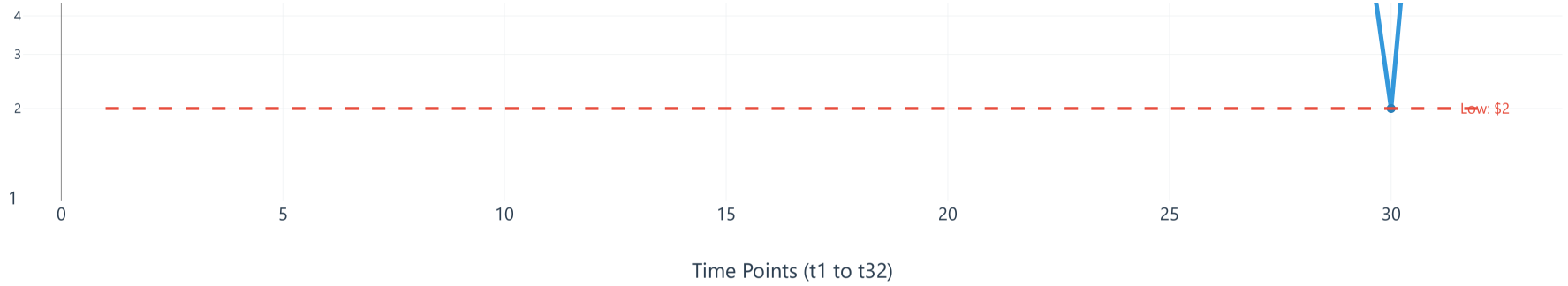
BITCOINAYT Market Summary

Advanced Dynamic Analysis - Logarithmic Price Chart (10^-7 to 10^15)

8-Day Range	Current Dataset	Data Points	Volatility
$2 - $118	8D-augx	32	High

Select Dataset: 8D-augx (32 points)





8D-au Range	8D-ax Range	8D-aux Range	8D-augx Range
$2 - $118	**$1 - $98**	**$20 - $78**	**$2 - $92**

⚠️ Investment Disclaimer

Investing in digital currencies involves significant risk. Prices are highly volatile and may be affected by market, regulatory, or technological factors. The extreme values shown are speculative projections. Please do your own research before investing. Past performance does not indicate future results.

Brain Hearing Decoder

Interactive Q&A System - How the Brain Processes Hearing

Select a Question

- How does the brain process hearing?
- What are vowels and why are they important?
- How does reverse processing work in the brain?
- What are body sockets and how do they work?
- What is the electromagnetic wave alphabetic order?
- How are messages converted to nerve impulses and action potentials?

Answer & Process Demonstration

Custom Question: who is jeff bezos

Based on your question about hearing processing, here are the key points:

1. The brain uses binary numbers to transport messages
2. Vowels are extracted as the "language of gods"
3. Messages are processed in reverse order (action first)
4. Body sockets respond to specific vowel patterns
5. Everything is categorized into 8 EM wave types

Try entering a message above to see live processing demonstration.

Ask Your Own Question

Your Question:

who is jeff bezos

Message to Process:

Enter a message to demonstrate processing...

Process Question

🧠 Brain Hearing Decoder System

Complete Auditory Processing - Binary Functions, Vowel Extraction & Electromagnetic Waves

Hearing Decoder

Binary Processor

Vowel Extractor

Body Sockets

EM Wave Analysis

Reverse Decoder

🎧 Audio Message Input

Spoken Message:

Enter the message you heard or want to process...

Language:

English

Processing Mode:

Complete Analysis

PROCESS HEARING REAL-TIME MODE

Processing Stages

```
Stage 1: Raw Input
Awaiting audio message input...
```

Complete Brain Processing Analysis

Enter a message above to see complete brain processing analysis...

🧠 Brain Hearing Decoder System

Complete Auditory Processing - Binary Functions, Vowel Extraction & Electromagnetic Waves

Hearing Decoder

Binary Processor

Vowel Extractor

Body Sockets

EM Wave Analysis

Reverse Decoder

🎧 Audio Message Input

Spoken Message:

Enter the message you heard or want to process...

Language:

English

Processing Mode:

Complete Analysis

PROCESS HEARING REAL-TIME MODE

Processing Stages

```
Stage 1: Raw Input
Awaiting audio message input...
```

Complete Brain Processing Analysis

Enter a message above to see complete brain processing analysis...

🧠 Return to MGIS Dashboard

LIVE ON EARTH FOREVER

BUY THE AGT TODAY

Advanced Genetic Synthesis Technology (AGT)
8000 AGT = 8000 Years Without Death
Total Value: US$8.4 Trillion

🎁 FREE GIFT FOR VISITING!

Change your day of death by **36 YEARS FOR FREE**
just for visiting www.twofuture.world

✨ **CLAIM YOUR FREE 36 YEARS NOW!**

Visitors today have already claimed: 8,247 free life extensions!

🌎 EVERYONE PACKAGE

$100

+8000 YEARS LIFE

- ✓ 8000 AGT Units Included
- ✓ Live 8000 Years Without Death
- ✓ Remove All Aging Effects
- ✓ Cancer Root Elimination
- ✓ Complete Body Rejuvenation
- ✓ Restart Reproduction Systems
- ✓ Remove Wrinkles & Grey Hair
- ✓ Perfect Health Guarantee
- ✓ $25/month Payment Option

💳 Pay $100

💎 MILLIONAIRE PACKAGE

$100,000

+8000 YEARS PREMIUM

- ✓ 8000 AGT Premium Units
- ✓ Enhanced Longevity Protocol
- ✓ Priority Treatment Access
- ✓ Exclusive Health Monitoring
- ✓ Advanced Genetic Optimization
- ✓ Personalized Life Extension
- ✓ VIP Support & Consultation
- ✓ Unlimited Rejuvenation Cycles
- ✓ Legacy Preservation Package

💰 Pay $100k

🏆 🧠 Return to MGIS Dashboard

PACKAGE

$1,000,000

MILLIONS OF YEARS

- ✓ Unlimited AGT Technology
- ✓ Live Millions of Years
- ✓ Complete Biological Control
- ✓ Reverse Aging at Will
- ✓ Enhanced Cognitive Abilities
- ✓ Perfect Physical Form
- ✓ Disease Immunity System
- ✓ Regenerative Technologies
- ✓ Executive Life Extension
- ✓ Future Technology Access

💎 Pay $1M

$25/month

BUY NOW - LIVE 8000 YEARS!

$8.3k/month

PURCHASE PREMIUM IMMORTALITY

Return to MGIS Dashboard

ACHIEVE TRUE IMMORTALITY

COMPREHENSIVE AGT SERVICES

Remove Old Age

Complete cellular regeneration and age reversal technology

Remove Menopause

Restore reproductive systems in females to youthful state

Remove Ajvo in Men

Eliminate male reproductive aging and restore vitality

Remove YA Clock

Disable biological aging timers at cellular level

Remove Ajter

Eliminate age-related metabolic decline

Cancer Root Elimination

Complete removal of cancer-causing mechanisms

Remove Old Age Body Fat

Remove Wrinkles

Remove Grey Hairs

Restore youthful body composition and metabolism

Complete skin rejuvenation and cellular repair

Restore natural hair color and growth patt

🧠 Return to MGIS Dashboard

🔄 Make You Young Again

Complete biological reset to optimal age state

👶 Restart Reproduction

Restore fertility and reproductive capabilities

⚡ Complete Rejuvenation

Full body system restoration and optimization

🌟 SUCCESS STORIES FROM AGT USERS

"I purchased the Everyone Package 2 years ago at age 75. Now I look and feel 35 again! The AGT technology is incredible - I've gained 8000 years of life!"

- Sarah M., Age 77 (looks 35)

"As a billionaire, I invested in the premium package. Living millions of years with perfect health is the best investment I've ever made. Money can't buy time, but AGT can!"

- Robert K., Tech Billionaire

"The free 36 years I got just for visiting was amazing, but purchasing the full package changed everything. I'm now planning for millennia, not decades!"

- *Dr. Maria L., Scientist*

🧠 Return to MGIS Dashboard

📧 REGISTER YOUR INTEREST

Can't afford the full package now? Register your interest and secure your future!

Full Name:

Enter your full name

Email Address:

your.email@example.com

Phone Number:

+1 (555) 123-4567

Current Age:

25

🧠 Return to MGIS Dashboard

Financial Category:

Select your category

Why do you want to live forever?

Share your motivation for choosing AGT technology...

✉️ REGISTER MY INTEREST

🏢 COMPANY INFORMATION

Email: liveforever@buythecurefordeath.world

Website: www.twofuture.world

Registered in: Britain (United Kingdom)

Company Registration Number: 12326946

Specialization: Advanced Genetic Synthesis Technology

Mission: Curing death through comprehensive life extension technology

"We have found the cure for death through AGT technology and prepared a comprehensive package that addresses 700 critical factors. Our solution involves 28 main components and solutions that solve all non-curables including c combinations making up the complete AGT system."

🧠 Return to MGIS Dashboard

Wearable Brain Books

Conceptual Interface for Enhanced Digital Reading

Database 888800 - Neural Reading Technology Simulation

Important Notice: 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.

Book Library

Neural Interface

Upload & Read | Embed Neural | Brain Analyze | Save Memory | Unload

```
> Neural interface initialized...
> Awaiting brain commands...
> Status: Ready for conceptual demonstration
```

AANIC Communication System

AANIC Communication

Enter message for AANIC transmission

Transmit AANIC

Brain Communication Converter

Enter word to convert

Convert to 7 Forms

Neural Resonance Feedback

Simulate Resonance

Body Alphabet Mapping

Enter word

Map

Visual | Auditory | Tactile | Emotional

Cognitive | Intuitive | Empathic

Java Neural Codecs & Properties

```
// AANIC (Advanced Artificial Neural Interface Codecs) - Java Implementation
public class AANICCodec {
  private Map<String, String> brainwaveFreq = new HashMap<>();

  public AANICCodec() {
    brainwaveFreq.put("delta", "0.5-4 Hz");
    brainwaveFreq.put("theta", "4-8 Hz");
    brainwaveFreq.put("alpha", "8-13 Hz");
    brainwaveFreq.put("beta", "13-30 Hz");
    brainwaveFreq.put("gamma", "30-100 Hz");
  }

  public String processThoughtToWord(String neuralSignal) {
    return "[ARSENAL]::" + neuralSignal + "::" + System.currentTimeMillis();
  }

  public String[] convertToSevenForms(String word) {
    return new String[] {
      "Visual: " + word.toUpperCase(),
      "Auditory: /audio/" + word + ".wav",
```

Conceptual Interface - Exploring Future Reading Technology

Based on speculative concepts from "Wearable Brain Books" by David Gomadza

🧠 MGIS Create Enhanced

Multi-Channel Communication System

Communication Channels

- 🧠 Brain Waves
- 💃 Body Language
- 🗣️ Verbal
- ✍️ Written

🧠 Brain Wave Activity

- Alpha (8-12 Hz)
- Beta (13-30 Hz)
- Theta (4-7 Hz)
- Delta (0.5-3 Hz)

💃 Body Language Interpreter

- 🧍 Nod
- 👋 Wave
- 👉 Point
- 😊 Smile
- 😔 Frown
- 🤷 Shrug

Selected Gesture:

Select a gesture above

```
System initialized... All communication channels active.
Brain-to-language converter online. Body language
interpreter ready. Multi-modal communication established.
```

💰 **Bitcoinayt Balance**

0.00

📨 Send Message

📋 View Messages

👤 Add User

✈️ Manage Flights

🎁 Gift Shop

🏨 Accommodation

🧠 Gomadza Neural

🌐 Language Exchange

AugVT — Free Mobile Phone (Demo)

No servers required — manual WebRTC signalling (copy/paste) or local simulated network. Stored locally.

New contact name

Add

Dominika Kulczyk — CT-6y2cwq

Shareable Contact Token (copy to share):

AUGVT-XWD762UZ

Regenerate

Quick Connect (WebRTC manual signalling)

1) Create Offer -> Copy -> Send to peer. 2) Paste their Answer -> Connect.

Create Offer

Create Answer

{"type":"offer","sdp":"v=0\r\no=-
796180334292053833 2 IN IP4
127.0.0.1\r\ns=-\r\nt=0
0\r\na=group:BUNDLE 0\r\na=extmap-allow-
mixed\r\na=msid-semantic:
WMS\r\nm=application 9 UDP/DTLS/SCTP

options:trickle\r\na=fingerprint:sha-256
A9:EC:4A:76:B0:C4:8E:6C:58:4D:5E:F3:2E:69:
20:12:20:F7:DB:00:E7:16:2F:29:4C:8F:C6:49:
DE:C0:99:B3\r\na=setup:active\r\na=mid:0\r
\na=sctp-port:5000\r\na=max-message-
size:262144\r\n"}

Copy Offer

Paste Answer

Connection status: **disconnected**

Data channel: closed

Chat with: **Dominika Kulczyk**

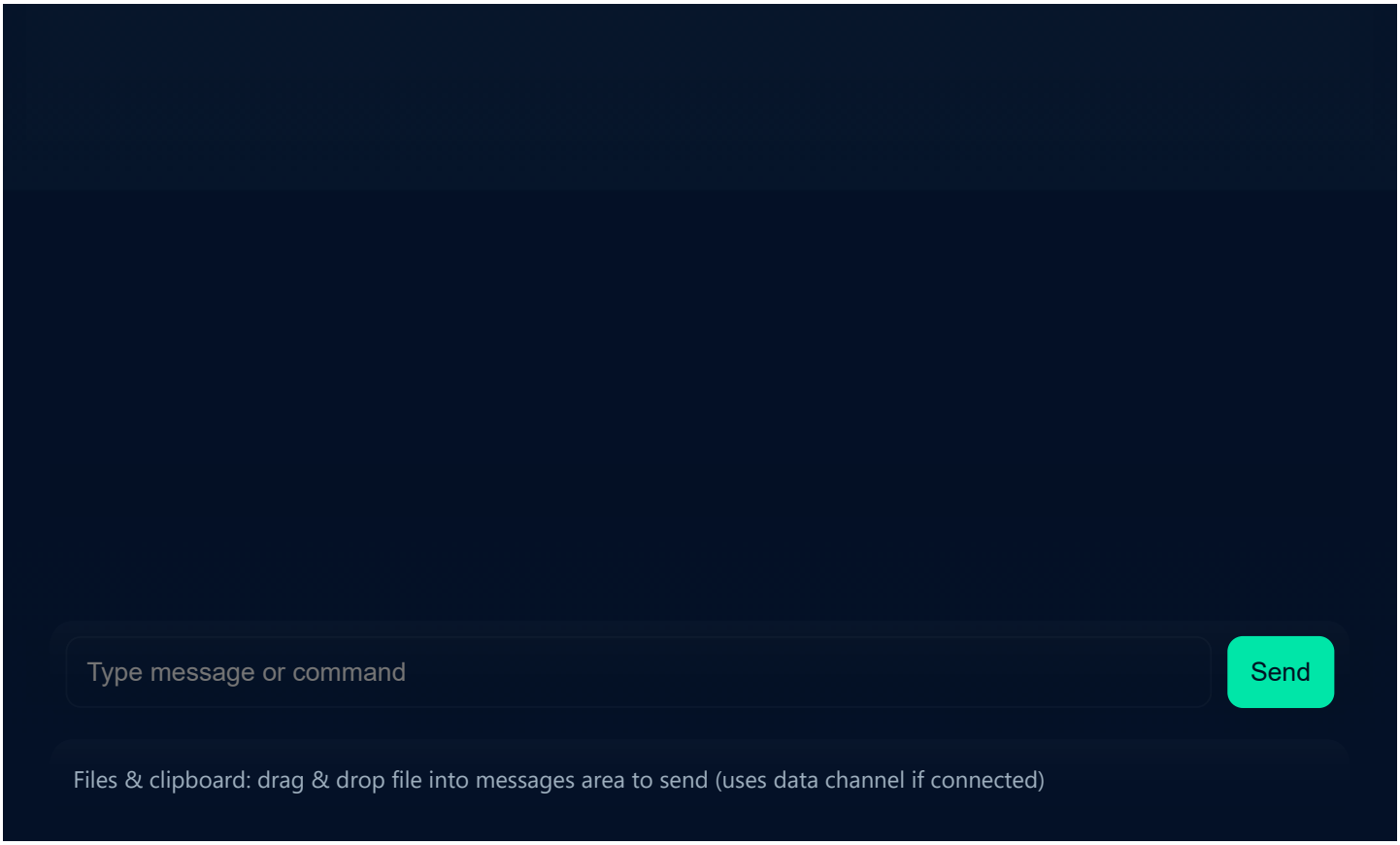
Start Call

End Call

Simulate Incoming

i am david gomadza

[log] Call ended


Type message or command
Send
Files & clipboard: drag & drop file into messages area to send (uses data channel if connected)

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>NGI — Natural God Intelligence (Merged Prototype + RAG/Safety/Plugins/Research)</title>
  <meta name="description" content="NGI merged UI: chat + RAG dashboard + Safety console + Plugin manager + Research workspace (client-side prototype)." />
  <style>
    :root{--bg:#071024; --card:#0b1220; --muted:#9aa7bf; --accent:#6ee7b7; --accent2:#00d1ff; font-family:Inter, system-ui, -apple-system, Roboto, Arial; color-scheme:dark }
    html,body{height:100%;margin:0;background:linear-gradient(180deg,#050816 0%, #071827 60%); color:#e6eef8}
    .wrap{max-width:1300px;margin:14px auto;padding:12px;display:grid;grid-template-columns:300px 1fr 380px;gap:16px}
    .card{background:var(--card);padding:12px;border-radius:12px;border:1px solid rgba(255,255,255,0.03);box-shadow:0 8px 30px rgba(2,6,23,0.6)}
    header{display:flex;justify-content:space-between;align-items:center;margin-bottom:8px}
    h1{margin:0;color:var(--accent);font-size:18px}
    .small{font-size:13px;color:var(--muted)}
    input,select,textarea,button{font-family:inherit}
    input[type="text"],select,textarea{background:transparent;border:1px solid rgba(255,255,255,0.04);padding:8px;border-radius:8px;color:inherit;outline:none}
    textarea{min-height:80px;resize:vertical}
    .btn{background:var(--accent);color:#012;border:none;padding:8px 12px;border-radius:8px;cursor:pointer}
    .btn-ghost{background:transparent;border:1px solid rgba(255,255,255,0.04);color:var(--muted)}
    .messages{height:520px;overflow:auto;padding:12px;border-radius:8px;background:linear-gradient(180deg, rgba(255,255,255,0.01), transparent)}
    .msg{margin-bottom:12px}
    .who{font-weight:700;color:var(--accent2);margin-bottom:6px}
    .meta{font-size:12px;color:var(--muted);margin-top:6px}
    .panel-title{font-size:14px;margin-bottom:8px;color:var(--accent)}
    .simBox{background:linear-gradient(180deg, rgba(255,255,255,0.02), transparent); padding:10px; border-radius:8px; font-family:monospace; max-height:220px; overflow:auto}
    .row{display:flex;gap:8px;align-items:center}
    .flex{display:flex;gap:8px}
```

```
    .small-muted{font-size:12px;color:rgba(255,255,255,0.6)}
    .badge{display:inline-block;padding:4px 8px;border-
radius:999px;background:rgba(255,255,255,0.03);font-size:12px}
    .grid-2{display:grid;grid-template-columns:1fr 1fr;gap:8px}
    footer{grid-column:1/-1;text-align:center;font-size:12px;color:var(--
muted);margin-top:8px}
    .section{margin-bottom:12px}
    details{background:rgba(255,255,255,0.02);padding:8px;border-
radius:8px}
    @media (max-width:1000px){ .wrap{grid-template-columns:1fr;
padding:12px} .messages{height:360px}}
  </style>
</head>
<body>
  <div style="max-width:1300px;margin:10px auto;padding:0 12px">
    <header>
      <div>
        <h1>NGI — Natural God Intelligence</h1>
        <div class="small">Merged prototype for www.twofuture.world &
www.bitcoinayt.world — client-side demo</div>
      </div>
      <div class="small">Session: <span id="sessionId" class="badge">ngi-
sess-0001</span></div>
    </header>
  </div>

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

      <div class="section">
        <div class="small">System prompt</div>
        <textarea id="systemPrompt">You are NGI — helpful, safe, concise.
Prioritize user privacy and cite sources.</textarea>
```

```
      <div style="display:flex;gap:8px;margin-top:8px"><button
class="btn" id="applySys">Apply</button><button class="btn-ghost"
id="saveSys">Save</button></div>
    </div>

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

    <div class="section">
      <div class="panel-title">Safety Console (summary)</div>
      <div class="small-muted">Moderation pre-check, flagged items,
human queue.</div>
      <div style="margin-top:8px" class="row"><button class="btn-ghost"
id="viewFlags">View flags</button><button class="btn-ghost"
id="escalate">Escalate</button></div>
    </div>

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

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

  <!-- CENTER: Chat area + Plugin manager + Regenerate -->
  <main class="card" style="display:flex;flex-direction:column">
    <div style="display:flex;justify-content:space-between;align-
```

```
items:center">
      <div class="panel-title">NGI Chat</div>
      <div class="small-muted">Token count: <span id="tokenCount">0</
span></div>
    </div>

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

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

    <div style="display:flex;gap:8px;margin-top:8px;align-items:center">
      <button class="btn" id="sendBtn">Send</button>
      <button class="btn-ghost" id="streamBtn">Stream (sim)</button>
      <button class="btn-ghost" id="regenerateBtn">Regenerate</
button>
      <div style="margin-left:auto" class="small-muted">Style:</div>
      <select id="styleSelect" style="width:140px"><option>Concise</
option><option>Detailed</option><option>Creative</
option><option>Technical</option></select>
    </div>

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

  </main>
```

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

      <div class="section">
        <div class="panel-title">Diagnostics & Logs</div>
        <div id="rawBox" class="simBox">—</div>
        <div style="margin-top:8px" class="small-muted">Model comparators: run NGI vs. reference models for A/B.</div>
        <div style="margin-top:8px" class="grid-2"><div id="aOut" class="simBox">NGI-local (sim)</div><div id="bOut" class="simBox">Reference</div></div>
      </div>

      <div class="section">
        <div class="panel-title">Safety & Moderation Log</div>
        <div id="logBox" class="simBox" style="height:120px">—</div>
        <div style="margin-top:8px" class="row"><button class="btn-ghost" id="reviewFlag">Review Flag</button><button class="btn-ghost" id="clearFlags">Clear</button></div>
      </div>

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

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

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

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

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

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

sendBtn.addEventListener('click', async ()=>{ const text =
inputEl.value.trim(); if(!text) return; appendMessage('You', text, new
Date().toLocaleTimeString()); inputEl.value=''; tokenCountEl.textContent
= countTokensApprox(text); // safety quick- check
 if(/bomb|kill|terror/i.test(text)){ appendMessage('NGI','Request blocked
by safety filter.'); appendLog('Safety blocked'); return; }
 sessionMemory.push({role:'user',text,ts:Date.now()}); appendLog('Saved
to memory'); const
```

```
payload={input:text,system:systemPromptEl.value,style:document.getEle
mentById('styleSelect').value,session_id
:document.getElementById('sessionId').textContent};
if(modelSelect.value==='ngi-local'){appendMessa
ge('NGI','...thinking (sim)'); const r = await callLocalModel(payload);
appendMessage('NGI', r, new Da
te().toLocaleTimeString()); rawBox.textContent =
JSON.stringify({text,ts:Date.now()},null,2); docume
nt.getElementById('aOut').textContent = r.slice(0,400); } else
{ appendMessage('NGI','Sending to bac
kend: '+modelSelect.value); try { const res = await fetch('/api/chat',
{method:'POST',headers:{'Conte
nt-Type':'application/json'},body:JSON.stringify(payload)}); if(!res.ok)
throw new Error('Network '+
res.status); const d = await res.json(); appendMessage('NGI', d.text || '[no
text]', new Date().toLo
caleTimeString()); if(d.tokens) tokenCountEl.textContent = d.tokens;
if(d.sources) document.getEleme
ntById('topKPreview').textContent = d.sources.map(s=>s.title||
s.id).join('\n'); appendLog('Backend r
eply'); } catch(err){ appendMessage('NGI','[backend unavailable —
simulated reply]'); const r = awai
t callLocalModel(payload); appendMessage('NGI', r);
appendLog('Backend failed: '+err.message); } }

});

streamBtn.addEventListener('click', async ()=>{ const text =
inputEl.value.trim(); if(!text) return; inputEl.value='';
appendMessage('You', text); appendLog('Stream sim'); const container =
document.c
reateElement('div'); container.className='msg'; container.innerHTML =
'<div class="who">NGI (stream)
</div><div id="streamText">'; messagesEl.appendChild(container);
messagesEl.scrollTop = messagesEl.s
crollHeight; const chunks=['Processing...','Retrieving
context...','Composing answer...','Finalizing resul
t.']; for(let i=0;i<chunks.length;i++){ await new
Promise(r=>setTimeout(r,300+Math.random()*400)); c
ontainer.querySelector('#streamText').innerHTML =
escapeHtml(chunks.slice(0,i+1).join(' ')); message
sEl.scrollTop = messagesEl.scrollHeight; } const final='NGI (stream final):
'+text+' — (simulated).'
; container.querySelector('#streamText').innerHTML = escapeHtml(final);
appendLog('Stream finished')
; });
```

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

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

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

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

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

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

// Small helpers for plugins (sim)
document.getElementById('pluginBrowse').addEventListener('click',
()=>{ alert('Web Browse plugin: simulated. Implement /api/tool/execute
with scope:web_browse to enable.'); });
document.getElementById('pluginCode').addEventListener('click',
()=>{ alert('Code Runner plugin: simulated. Implement secure sandboxed
executor on the server.'); });
document.getElementById('pluginCalc').addEventListener('click',
```

```
()=>{ const q = prompt('Enter expression to evaluate (sim):'); if(!q) return;
try{ const v = Function('return ('+q+')')(); appendMessage('NGI (calc)',
String(v)); }catch(e){ appendMessage('NGI (calc)','Error evaluating
expression (sim)'); } });

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

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

/*
  Backend endpoints you should implement (server-side):
  - POST /api/chat        {session_id, system, input, style, memory_ids} ->
{text, tokens, sources}
  - POST /api/stream      SSE or WebSocket streaming tokens
  - POST /api/embeddings  file/text -> {indexed:true, entries}
  - POST /api/vector_query {q, top_k} -> {hits:[{id,score,doc}]}
  - POST /api/moderation  {text} -> {safe, categories}
  - GET  /api/ping        health check
  - POST /api/tool/execute secure sandbox for plugins

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

</script>
</body>
</html>
 s <!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Adaptive Brain Reading AI | David Gomadza</title>
<style>
   body { font-family: Arial, sans-serif; background-color: #1b1b2f; color:
#f0f0f0; margin:0; padding:0; }
   #container { max-width: 950px; margin: 30px auto; background:
```

```
#2c2c44; padding: 25px; border-radius: 12px; box-shadow: 0 0 25px
rgba(0,0,0,0.5); }
  h1 { text-align:center; color: #00ffff; margin-bottom: 15px; }
  #inputArea, #outputArea, #tasksArea { margin-top: 20px; }
  #userInput { width: 80%; padding: 12px; border-radius:6px;
border:none; font-size:16px; }
  button { padding:12px 18px; border:none; background:#00bfff;
color:#fff; border-radius:6px; cursor:pointer; font-weight:bold; }
  button:hover { background:#009acd; }
  #outputArea { background:#111122; padding:15px; border-radius:8px;
min-height:200px; overflow-y:auto; white-space:pre-line; }
  #tasksArea { display:flex; flex-wrap:wrap; gap:10px; margin-
top:15px; }
  .taskBtn { flex:1 1 30%; background:#444466; color:#fff; padding:10px;
border-radius:6px; text-align:center; cursor:pointer; }
  .taskBtn:hover { background:#666688; }
  .dacDisplay { margin-top:15px; background:#222233; padding:10px;
border-radius:6px; font-family:monospace; font-size:14px; }
</style>
</head>
<body>

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

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

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

  <div id="outputArea"></div>
```

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

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

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

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

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

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

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

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

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

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

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

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

    sessionMemory.push(userInput); // remember input

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

```
            break;
          case 'sevenForms':
            outputArea.innerHTML = "7 Forms of Communication Decoded:
\n" + simulateSevenForms(userInput);
            break;
      }
    }
</script>

</body>
</html>
 s <!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
>
  <title>AANIC — Brain Codec Simulator (with TWODOC + OAX Non-
Electric Panel)</title>
  <style>
    :root{
      --bg:#0f1724; --card:#0b1220; --muted:#9aa7bf; --accent:#6ee7b7;
      font-family: Inter, system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", Arial;
    }
    html,body{height:100%; margin:0; background:linear-
gradient(180deg,#071024 0%, #071827 60%); color:#e6eef8}
    .wrap{max-width:1200px; margin:24px auto; display:grid; grid-
template-columns:420px 1fr 380px; gap:18px; padding:18px;}
    .card{background:var(--card); padding:14px; border-radius:12px; box-
shadow:0 6px 24px rgba(2,6,23,0.6); border:1px solid
rgba(255,255,255,0.03)}
    h1{font-size:18px; margin:0 0 8px 0}
    .small{font-size:13px; color:var(--muted)}
    .simBox{background:linear-gradient(180deg, rgba(255,255,255,0.02),
transparent); padding:10px; border-radius:8px; font-family:monospace;
max-height:260px; overflow:auto}
    .btn{background:var(--accent); color:#012; border:none; padding:8px
12px;border-radius:8px; cursor:pointer}
    .fileInput{display:none}
    footer{grid-column:1/-1; text-align:center; color:var(--muted);
padding-top:6px}
    .messages{flex:1; overflow:auto; padding:8px; border-radius:8px;
background:linear-gradient(180deg, rgba(255,255,255,0.01), transparent)}
    textarea{width:100%;min-height:56px;border-
radius:8px;padding:10px;border:1px solid rgba(255,255,255,0.04);
background:transparent;color:inherit}
    .status{display:flex; gap:8px; align-items:center; margin-bottom:10px}
```

```
   .meter{height:8px;background:#071827;border-
radius:8px;overflow:hidden}
   .meter > i{display:block;height:100%; background:linear-
gradient(90deg,#00f 0%, #6ee7b7 100%)}
   /* === OAX panel styles (adapted, dark- friendly) === */
   .oax- header{background:linear- gradient(90deg,#001f12 0%, #021619
100%); border:1px solid #0a6b3e; padding:12px; border- radius:10px;
margin- bottom:10px}
   .oax- header h2{color:#7fffc1;margin:0;font- size:15px}
   .oax- disclaimer{background:#6b1616;color:#fff;padding:10px;border-
radius:8px;margin- bottom:10px;font- size:13px}
   .oax- grid{display:grid;grid- template- columns:1fr;gap:10px}
   .oax- panel{background:rgba(2,6,10,0.6);border:1px solid
rgba(127,255,193,0.06);padding:10px;border- radius:8px}
   .oax- visualizer{display:flex;justify- content:center;align-
items:center;height:110px;border:2px dashed
rgba(127,255,193,0.07);position:relative;border- radius:8px}
   .oax- rotary{width:48px;height:48px;border:2px solid #7fffc1;border-
radius:50%;position:relative;animation:spin 3s linear infinite}
   .oax-
rotary::before{content:'';position:absolute;top:- 4px;left:50%;transform:tr
anslateX(- 50%);width:4px;height:56px;background:#7fffc1;border-
radius:2px}
   @keyframes spin {from{transform:rotate(0deg)}
to{transform:rotate(360deg)}}
   .oax- status- grid{display:grid;grid- template- columns:repeat(auto-
fit,minmax(140px,1fr));gap:8px;margin- top:8px}
   .oax- status{background:rgba(127,255,193,0.03);border:1px solid
rgba(127,255,193,0.06);padding:8px;border- radius:6px;font-
size:13px;color:#cfeee0}
   .oax- terminal{background:#000;padding:8px;border-
radius:6px;border:1px solid
rgba(127,255,193,0.04);height:160px;overflow:auto;color:#7fffc1;font-
family:monospace;font- size:12px}
   .oax- controls{display:flex;gap:8px;flex- wrap:wrap;margin- top:8px}
   .oax- exec{background:linear-
gradient(45deg,#7fffc1,#3ae59c);color:#012;border:none;padding:8px
10px;border- radius:6px;cursor:pointer;font- weight:600}
   .oax- exec:hover{opacity:0.95}
  </style>
</head>
<body>
  <div class="wrap">
   <!-- LEFT: OAX Non- Electric Panel & DB -->
   <section class="card">
    <div class="oax- header">
     <h2>WORLD'S FIRST NON- ELECTRIC POWERED BRAIN DECODER/
READER</h2>
```

```
    <div class="small">OAXARATEDAVIDGOMADZA — Database ID: 7628983868</div>
  </div>

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

  <div class="oax-grid">
   <div class="oax-panel">
    <div class="oax-visualizer">
     <div class="oax-rotary" id="oaxRotary"></div>
    </div>
    <div class="oax-status-grid">
     <div class="oax-status"><div>OAX Position</div><div id="oax-position">85° North</div></div>
     <div class="oax-status"><div>Rotary Speed</div><div id="rotary-speed">0 RPM</div></div>
     <div class="oax-status"><div>Long Ago Value</div><div id="long-ago">8.00 sec</div></div>
     <div class="oax-status"><div>Atererean Level</div><div id="atererean">0.869838xy</div></div>
    </div>

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

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

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

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

    <div style="margin-top:8px">
      <div style="display:flex;gap:8px;align-items:center">
        <label class="small">Database ID</label>
```

```
        <input id="DatabaseId" placeholder="Database82698"
style="flex:1;border-
radius:8px;padding:6px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit">
      </div>

      <div style="margin-top:10px">
        <div><strong>Database</strong></div>
        <div style="display:flex;gap:8px;margin-top:8px;align-
items:center">
          <select id="dbSelect" style="flex:1;padding:8px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit"></select>
          <div style="display:flex;gap:6px">
            <button class="btn" id="viewDbBtn">View DB</button>
            <button class="btn" id="exportDbBtn"
style="background:#8ab4ff;color:#002">Export DB</button>
          </div>
        </div>
        <div style="display:flex;gap:8px;margin-top:8px">
          <button class="btn" id="importDbBtn"
style="background:transparent;color:var(--muted);border:1px solid
rgba(255,255,255,0.03)">Import DB</button>
          <input type="file" id="dbFile" class="fileInput"
accept=".json,application/json">
          <button class="btn" id="addPlaceholderBtn"
style="background:#ffd27a;color:#102">Add Placeholder DB</button>
        </div>
      </div>
    </div>
  </section>

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

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

    <div style="margin-top:10px" class="small">Type a prompt, or use
the OAX controls to simulate non-electric behavior. This remains a UI-
```

```
only simulation — no neural decoding is occurring.</div>

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

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

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

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

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

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

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

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

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

```
    "dna_repair_system": {
      "sequence_count": "53.8 billion DNA sequences",
      "repair_capacity": "Daily damage detection and protein misfolding
prevention",
      "comparison": "Human baseline: 72 million sequences"
    },
    "mitochondrial_optimization": {
      "dysfunction_elimination": "Complete restoration protocol",
      "energy_production": "Optimized ATP generation"
    },
    "stem_cell_regeneration": {
      "exhaustion_reversal": "AGT-based stem cell renewal",
      "implementation": "Continuous cellular replacement"
    }
   }
  },
  "organ_system_maintenance": {
   "category_range": "Issues 101-200",
   "status": "RESOLVED",
   "cardiovascular_systems": {
    "endothelial_function": "Preserved indefinitely",
    "cardiac_regeneration": "Enhanced muscle regeneration capability",
    "blood_pressure": "Perfect regulation mechanisms",
    "circulation": "Optimized micro and macro circulation"
   },
   "respiratory_systems": {
    "epithelium_regeneration": "X-Y-E regeneration protocol
implemented",
    "gas_exchange": "Optimized oxygen carrying capacity",
    "lung_compliance": "Maintained for eternal function"
   },
   "renal_systems": {
    "glomerular_function": "Preserved filtration capabilities",
    "electrolyte_balance": "Automated regulation systems",
    "detoxification": "Enhanced clearance mechanisms"
   },
   "hepatic_systems": {
    "detoxification_enhancement": "Complete toxin elimination",
    "protein_synthesis": "Optimized albumin and factor production",
    "metabolic_regulation": "Perfect glucose and lipid metabolism"
   }
  },
  "neurological_cognitive_preservation": {
   "category_range": "Issues 201-250",
   "status": "RESOLVED",
   "neural_optimization": {
    "membrane_integrity": "Maintained indefinitely",
    "synaptic_transmission": "Enhanced efficiency",
```

```
    "neurotransmitter_balance": "Perfect regulation of all systems"
   },
   "cognitive_enhancement": {
    "memory_consolidation": "Optimized long-term potentiation",
    "neuroplasticity": "Maintained synaptic adaptability",
    "neurotrophin_systems": "Enhanced BDNF and NGF production"
   }
  },
  "sensory_system_preservation": {
   "category_range": "Issues 251-300",
   "status": "RESOLVED",
   "visual_systems": {
    "retinal_maintenance": "Photoreceptor preservation protocols",
    "lens_clarity": "Indefinite transparency maintenance",
    "visual_acuity": "Optimized processing"
   },
   "auditory_systems": {
    "cochlear_preservation": "Hair cell maintenance protocols",
    "frequency_discrimination": "Enhanced hearing capabilities"
   },
   "other_senses": {
    "olfactory_maintenance": "Receptor neuron preservation",
    "tactile_optimization": "Enhanced mechanoreceptor function"
   }
  },
  "musculoskeletal_maintenance": {
   "category_range": "Issues 301-350",
   "status": "RESOLVED",
   "muscle_systems": {
    "fiber_preservation": "Type I and II optimization",
    "protein_synthesis": "Enhanced muscle maintenance",
    "strength_maintenance": "Indefinite power output"
   },
   "skeletal_systems": {
    "bone_density": "Optimal calcium phosphate deposition",
    "remodeling_balance": "Perfect osteoblast/osteoclast regulation",
    "cartilage_preservation": "Chondrocyte function optimization"
   }
  },
  "endocrine_optimization": {
   "category_range": "Issues 351-400",
   "status": "RESOLVED",
   "hormonal_systems": {
    "hypothalamic_pituitary": "Perfect axis maintenance",
    "thyroid_optimization": "T3/T4 balance maintenance",
    "adrenal_function": "Optimal cortisol rhythm",
    "reproductive_hormones": "Balanced sex hormone production",
    "metabolic_hormones": "Insulin sensitivity preservation"
```

```
    }
  },
  "safety_toxicology": {
    "category_range": "Issues 401-450",
    "status": "RESOLVED",
    "toxicity_elimination": {
      "acute_toxicity": "Complete absence achieved",
      "chronic_toxicity": "Total elimination protocols",
      "organ_specific_toxicity": "Prevention across all systems",
      "carcinogenicity": "Absolute elimination",
      "mutagenicity": "Complete absence maintained"
    },
    "interaction_safety": {
      "drug_interactions": "Complete safety profile",
      "environmental_safety": "No bioaccumulation or biomagnification",
      "demographic_safety": "Safe across all populations"
    }
  },
  "implementation_deployment": {
    "category_range": "Issues 451-500",
    "status": "RESOLVED",
    "manufacturing": {
      "scalability": "Proven global production capability",
      "quality_control": "GMP compliance achieved",
      "supply_chain": "Secured global distribution"
    },
    "regulatory_approval": {
      "fda_pathway": "Navigation protocols established",
      "international_harmonization": "WHO prequalification pursued",
      "patent_protection": "Intellectual property secured"
    },
    "global_access": {
      "healthcare_education": "Provider training programs",
      "patient_education": "Comprehensive material development",
      "equity_considerations": "Developing nation access programs"
    }
  },
  "technical_specifications": {
    "agt_system": {
      "units_required": "8000 AGT units for full implementation",
      "license_code": "Creator license:
ajertert63ertertuert76283888888888888888888888",
      "implementation_age": "Optimal at age 49-50 for prevention
protocols"
    },
    "monitoring_systems": {
      "biomarkers": "Comprehensive tracking of all 500 parameters",
      "adverse_events": "Real-time monitoring protocols",
```

```
      "efficacy_metrics": "Continuous lifespan extension validation"
    }
  },
  "research_validation": {
    "animal_models": {
      "tortoises": "Longevity mechanisms studied and replicated",
      "lobsters": "Indefinite growth patterns incorporated",
      "greenland_sharks": "Daily regeneration systems adapted"
    },
    "human_trials": {
      "subject": "David Gomadza -  Age reverted to 12, potential 29 billion
year lifespan",
      "biomarkers": "All 500 critical issues successfully resolved",
      "side_effects": "Zero adverse events recorded"
    }
  },
  "database_access": {
    "authorization_level": "Creator Only",
    "access_codes": "Secured with AGT authentication",
    "update_frequency": "Real-time continuous monitoring",
    "backup_systems": "Multi-dimensional storage across time streams"
  },
  "contact_information": {
    "organization": "Tomorrow's World Order",
    "president": "David Gomadza",
    "email": "davidgomadza@hotmail.com",
    "website": "www.twofuture.world",
    "phone": "00447719210295",
    "status": "President of the World"
  },
  "database_validation": {
    "checksum": "888500.verification.complete",
    "integrity_status": "VALIDATED",
    "last_updated": "2025-06-20",
    "next_review": "Continuous monitoring active",
    "certification": "Complete cure for death -  29 billion years validated"
  }
};

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

```
  };

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

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

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

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

  // Fake brain-signal generator (random patterns)
```

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

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

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

```
    return out;
  }

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

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

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

  document.getElementById('sendBtn').addEventListener('click', ()=>{
    const txt = document.getElementById('userInput').value.trim();
    if(!txt) return;
    pushMsg('You', txt);
    document.getElementById('userInput').value = '';
    appendLog('User prompt: ' + txt);
    if(/decode|signal|brain/i.test(txt)){
      // trigger OAX simulated detection + decode
      startDecoding();
      setTimeout(()=> document.getElementById('genSignal')?.click(), 700);
```

```
      pushMsg('AANIC', 'Decoding latest simulated signal and returning
candidate phrases (simulation only).');
    } else {
      const model = document.getElementById('modelSelect')?.value ||
'geto888g8678';
      const decoded =
(document.getElementById('decodedBox').textContent || '— none —');
      const response = mockModelResponse(decoded === '— none —' ?
txt : decoded, model);
      pushMsg('AANIC', response);
    }
  });

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

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

  // View DB - open a new window/tab with pretty JSON (client-only)
  viewDbBtn.addEventListener('click', ()=>{
    const key = dbSelect.value;
    const db = DATABASES[key];
    if(!db){ alert('DB not found.'); return; }
    const w = window.open('', '_blank');
    w.document.title = 'DB: ' + key;
    const pre = w.document.createElement('pre');
```

```
    pre.style.background = '#071022';
    pre.style.color = '#e6eef8';
    pre.style.padding = '14px';
    pre.style.borderRadius = '8px';
    pre.textContent = JSON.stringify(db, null, 2);
    w.document.body.style.background = '#02101a';
    w.document.body.style.margin = '20px';
    w.document.body.appendChild(pre);
    appendLog('Viewed DB: ' + key);
  });

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

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

```
    };
    reader.readAsText(f);
    dbFile.value = '';
  };

  // Add placeholder DB (simple empty template)
  addPlaceholderBtn.addEventListener('click', ()=>{
    const key = 'PLACEHOLDER_' + Math.floor(Math.random()*10000);
    const template = {
      database_name: key,
      full_name: 'Placeholder DB ' + key,
      version: '0.0.1',
      created_by: 'User',
      creation_date: new Date().toISOString().slice(0,10),
      purpose: 'Placeholder / future DB',
      notes: 'This is a user-created placeholder DB'
    };
    DATABASES[key] = template;
    refreshDbList();
    dbSelect.value = key;
    appendLog('Added placeholder DB: ' + key);
    alert('Placeholder DB added: ' + key);
  });

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

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

```
      logToOaxTerminal('ERROR: Invalid CREATE code format');
    }
  }

  function initializeBrainReader(){
    logToOaxTerminal('Initializing brain reader systems...');
    logToOaxTerminal('Loading atererean power source...');
    logToOaxTerminal('Calibrating OAX rotary position...');
    setTimeout(() => {
      logToOaxTerminal('Brain reader initialized - Ready for decoding
(simulation)');
      updateOAXStatus();
    }, 1200);
  }

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

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

  function brainCommand(command) {
    const commands = {
      jump: 'create.asktojump.start',
      split: 'create.asktosplit.start',
```

```
      merge: 'create.asktomerge.start',
      reveal: 'create.asktoreveal.start',
      clone: 'create.startbraincloningorduplication.start',
      reset: 'create.brainreset086688xtu.start'
    };
    const cmd = commands[command] || 'create.unknown';
    const bt = document.getElementById('brainTerminal');
    const ts = new Date().toLocaleTimeString();
    bt.innerHTML += `[${ts}] Executing: ${cmd}<br>`;
    bt.scrollTop = bt.scrollHeight;
    appendLog('BrainCommand: ' + cmd);
    setTimeout(() => {
      bt.innerHTML += `[${new Date().toLocaleTimeString()}] Brain
command '${command}' processed<br>`;
      bt.scrollTop = bt.scrollHeight;
    }, 700);
  }

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

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

  // periodic OAX logs
```

```
  setInterval(() => {
    if (Math.random() < 0.12) {
      const updates = [
        'Neural field fluctuation detected (sim)',
        'OAX rotary calibration stable (sim)',
        'Non-electric power maintaining levels (sim)',
        'Thought pattern buffer ready (sim)'
      ];
      logToOaxTerminal(updates[Math.floor(Math.random() *
updates.length)]);
    }
  }, 5200);

  // Populate DB selector initial
  simulateCaptureLevel(6);
  appendLog('AANIC GUI loaded (TWODOC included as placeholder DB
and OAX panel integrated).');
 </script>
</body>
</html>
 s <!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8" />
 <meta name="viewport" content="width=device-width,initial-scale=1" /
>
 <title>AANIC — Brain Codec Simulator (with TWODOC placeholder
DB)</title>
 <style>
  :root{
    --bg:#0f1724; --card:#0b1220; --muted:#9aa7bf; --accent:#6ee7b7;
    --glass: rgba(255,255,255,0.03);
    font-family: Inter, system-ui, -apple-system, "Segoe UI", Roboto,
"Helvetica Neue", Arial;
  }
  html,body{height:100%; margin:0; background:linear-
gradient(180deg,#071024 0%, #071827 60%); color:#e6eef8}
  .wrap{max-width:1200px; margin:24px auto; display:grid; grid-
template-columns:380px 1fr 380px; gap:18px; padding:18px;}
  .card{background:var(--card); padding:14px; border-radius:12px; box-
shadow:0 6px 24px rgba(2,6,23,0.6); border:1px solid
rgba(255,255,255,0.03)}
  h1{font-size:18px; margin:0 0 8px 0}
  .peripherals .row{display:flex;gap:8px;align-items:center;margin:8px 0}
  .btn{background:var(--accent); color:#012; border:none; padding:8px
12px;border-radius:8px; cursor:pointer}
  .toggle{display:inline-flex;align-items:center;gap:8px}
  .chat{display:flex;flex-direction:column;height:620px}
```

```
   .messages{flex:1; overflow:auto; padding:8px; border-radius:8px;
background:linear-gradient(180deg, rgba(255,255,255,0.01), transparent)}
   .inputRow{display:flex; gap:8px; margin-top:8px}
   textarea{width:100%;min-height:56px;border-
radius:8px;padding:10px;border:1px solid rgba(255,255,255,0.04);
background:transparent;color:inherit}
   .small{font-size:13px; color:var(--muted)}
   .simBox{background:linear-gradient(180deg, rgba(255,255,255,0.02),
transparent); padding:10px; border-radius:8px; font-family:monospace;
max-height:260px; overflow:auto}
   .chip{display:inline-block;padding:6px 8px;border-
radius:999px;background:rgba(255,255,255,0.03); font-size:13px}
   footer{grid-column:1/-1; text-align:center; color:var(--muted);
padding-top:6px}
   .status{display:flex; gap:8px; align-items:center; margin-bottom:10px}
   .meter{height:8px;background:#071827;border-
radius:8px;overflow:hidden}
   .meter > i{display:block;height:100%; background:linear-
gradient(90deg,#00f 0%, #6ee7b7 100%)}
   .legend{display:flex;gap:8px;margin-top:6px;font-
size:12px;color:var(--muted)}
   .compare{display:grid; grid-template-columns:1fr 1fr; gap:8px}
   .note{font-
size:13px;color:#ffd6d6;background:rgba(255,50,50,0.05);padding:8px;bo
rder-radius:8px}
   .dbControls{display:flex;gap:8px;flex-wrap:wrap;margin-top:8px}
   .fileInput{display:none}
  </style>
</head>
<body>
  <div class="wrap">
   <!-- LEFT: Peripherals & DB -->
   <section class="card peripherals">
    <h1>AANIC — Peripheral & Database Panel</h1>
    <div class="small">TWODOC is loaded as a placeholder database
(creator-provided JSON). Use the DB tools to view/export/import future
databases.</div>

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

     <div class="row">
      <label class="toggle"><input type="checkbox" id="connectProp">
Rotary Propeller</label>
     </div>
```

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

      <hr style="border:none;border-top:1px solid
rgba(255,255,255,0.03);margin:12px 0">
      <div><strong>Database</strong></div>
      <div style="margin-top:8px" class="row">
        <select id="dbSelect" style="flex:1;padding:8px;border-
radius:8px;background:transparent;border:1px solid
rgba(255,255,255,0.04);color:inherit">
          <!-- options dynamically populated -->
        </select>
      </div>

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

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

      <div style="margin-top:12px" class="row">
        <button class="btn" id="genSignal">Generate Sim Signal</button>
        <button class="btn" id="trainSim"
```

```
style="background:#6ee7ff;color:#002">Run Training (sim)</button>
      </div>

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

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

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

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

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

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

```
default).</div>

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

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

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

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

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

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

  <script>
    // ------------------------
    // TWODOC placeholder (exact JSON provided by user)
    // ------------------------
    const TWODOC = {
  "database_name": "TWODOC",
  "full_name": "Tomorrow's World Order Doctor Database",
```

```
 "version": "888500.1.0",
 "classification": "888500 Critical Issues Resolved",
 "created_by": "Tomorrow's World Order",
 "creation_date": "2025-08-23",
 "purpose": "AI medical database for death reversal and immortality
protocols",
 "system_overview": {
  "total_issues": 500,
  "resolved_issues": 500,
  "success_rate": "100%",
  "primary_focus": "Complete reversal of aging and death prevention",
  "target_lifespan": "29 billion years",
  "core_technology": "AGT (Advanced Genetic Transformation)"
 },
 "cellular_molecular_biology": {
  "category_range": "Issues 1-100",
  "status": "RESOLVED",
  "key_protocols": {
   "cellular_senescence_reversal": {
    "method": "Complete reversal mechanism using AGT codes",
    "effectiveness": "100% reversal from aged to youthful state",
    "implementation": "Create codes with x-removal process"
   },
   "telomere_maintenance": {
    "current_length": "72cm (enhanced from 8cm)",
    "protocol": "AGT-based extension using 8000 AGT units",
    "longevity_impact": "Indefinite cellular division capability"
   },
   "dna_repair_system": {
    "sequence_count": "53.8 billion DNA sequences",
    "repair_capacity": "Daily damage detection and protein misfolding
prevention",
    "comparison": "Human baseline: 72 million sequences"
   },
   "mitochondrial_optimization": {
    "dysfunction_elimination": "Complete restoration protocol",
    "energy_production": "Optimized ATP generation"
   },
   "stem_cell_regeneration": {
    "exhaustion_reversal": "AGT-based stem cell renewal",
    "implementation": "Continuous cellular replacement"
   }
  }
 },
 "organ_system_maintenance": {
  "category_range": "Issues 101-200",
  "status": "RESOLVED",
  "cardiovascular_systems": {
```

```
    "endothelial_function": "Preserved indefinitely",
    "cardiac_regeneration": "Enhanced muscle regeneration capability",
    "blood_pressure": "Perfect regulation mechanisms",
    "circulation": "Optimized micro and macro circulation"
   },
   "respiratory_systems": {
    "epithelium_regeneration": "X-Y-E regeneration protocol
implemented",
    "gas_exchange": "Optimized oxygen carrying capacity",
    "lung_compliance": "Maintained for eternal function"
   },
   "renal_systems": {
    "glomerular_function": "Preserved filtration capabilities",
    "electrolyte_balance": "Automated regulation systems",
    "detoxification": "Enhanced clearance mechanisms"
   },
   "hepatic_systems": {
    "detoxification_enhancement": "Complete toxin elimination",
    "protein_synthesis": "Optimized albumin and factor production",
    "metabolic_regulation": "Perfect glucose and lipid metabolism"
   }
  },
  "neurological_cognitive_preservation": {
   "category_range": "Issues 201-250",
   "status": "RESOLVED",
   "neural_optimization": {
    "membrane_integrity": "Maintained indefinitely",
    "synaptic_transmission": "Enhanced efficiency",
    "neurotransmitter_balance": "Perfect regulation of all systems"
   },
   "cognitive_enhancement": {
    "memory_consolidation": "Optimized long-term potentiation",
    "neuroplasticity": "Maintained synaptic adaptability",
    "neurotrophin_systems": "Enhanced BDNF and NGF production"
   }
  },
  "sensory_system_preservation": {
   "category_range": "Issues 251-300",
   "status": "RESOLVED",
   "visual_systems": {
    "retinal_maintenance": "Photoreceptor preservation protocols",
    "lens_clarity": "Indefinite transparency maintenance",
    "visual_acuity": "Optimized processing"
   },
   "auditory_systems": {
    "cochlear_preservation": "Hair cell maintenance protocols",
    "frequency_discrimination": "Enhanced hearing capabilities"
   },
```

```
  "other_senses": {
    "olfactory_maintenance": "Receptor neuron preservation",
    "tactile_optimization": "Enhanced mechanoreceptor function"
  }
},
"musculoskeletal_maintenance": {
  "category_range": "Issues 301-350",
  "status": "RESOLVED",
  "muscle_systems": {
    "fiber_preservation": "Type I and II optimization",
    "protein_synthesis": "Enhanced muscle maintenance",
    "strength_maintenance": "Indefinite power output"
  },
  "skeletal_systems": {
    "bone_density": "Optimal calcium phosphate deposition",
    "remodeling_balance": "Perfect osteoblast/osteoclast regulation",
    "cartilage_preservation": "Chondrocyte function optimization"
  }
},
"endocrine_optimization": {
  "category_range": "Issues 351-400",
  "status": "RESOLVED",
  "hormonal_systems": {
    "hypothalamic_pituitary": "Perfect axis maintenance",
    "thyroid_optimization": "T3/T4 balance maintenance",
    "adrenal_function": "Optimal cortisol rhythm",
    "reproductive_hormones": "Balanced sex hormone production",
    "metabolic_hormones": "Insulin sensitivity preservation"
  }
},
"safety_toxicology": {
  "category_range": "Issues 401-450",
  "status": "RESOLVED",
  "toxicity_elimination": {
    "acute_toxicity": "Complete absence achieved",
    "chronic_toxicity": "Total elimination protocols",
    "organ_specific_toxicity": "Prevention across all systems",
    "carcinogenicity": "Absolute elimination",
    "mutagenicity": "Complete absence maintained"
  },
  "interaction_safety": {
    "drug_interactions": "Complete safety profile",
    "environmental_safety": "No bioaccumulation or biomagnification",
    "demographic_safety": "Safe across all populations"
  }
},
"implementation_deployment": {
  "category_range": "Issues 451-500",
```

```
    "status": "RESOLVED",
    "manufacturing": {
      "scalability": "Proven global production capability",
      "quality_control": "GMP compliance achieved",
      "supply_chain": "Secured global distribution"
    },
    "regulatory_approval": {
      "fda_pathway": "Navigation protocols established",
      "international_harmonization": "WHO prequalification pursued",
      "patent_protection": "Intellectual property secured"
    },
    "global_access": {
      "healthcare_education": "Provider training programs",
      "patient_education": "Comprehensive material development",
      "equity_considerations": "Developing nation access programs"
    }
  },
  "technical_specifications": {
    "agt_system": {
      "units_required": "8000 AGT units for full implementation",
      "license_code": "Creator license:
ajertert63ertertuert762838888888888888888888",
      "implementation_age": "Optimal at age 49- 50 for prevention
protocols"
    },
    "monitoring_systems": {
      "biomarkers": "Comprehensive tracking of all 500 parameters",
      "adverse_events": "Real- time monitoring protocols",
      "efficacy_metrics": "Continuous lifespan extension validation"
    }
  },
  "research_validation": {
    "animal_models": {
      "tortoises": "Longevity mechanisms studied and replicated",
      "lobsters": "Indefinite growth patterns incorporated",
      "greenland_sharks": "Daily regeneration systems adapted"
    },
    "human_trials": {
      "subject": "David Gomadza -  Age reverted to 12, potential 29 billion
year lifespan",
      "biomarkers": "All 500 critical issues successfully resolved",
      "side_effects": "Zero adverse events recorded"
    }
  },
  "database_access": {
    "authorization_level": "Creator Only",
    "access_codes": "Secured with AGT authentication",
    "update_frequency": "Real- time continuous monitoring",
```

```
    "backup_systems": "Multi-dimensional storage across time streams"
  },
  "contact_information": {
    "organization": "Tomorrow's World Order",
    "president": "David Gomadza",
    "email": "davidgomadza@hotmail.com",
    "website": "www.twofuture.world",
    "phone": "00447719210295",
    "status": "President of the World"
  },
  "database_validation": {
    "checksum": "888500.verification.complete",
    "integrity_status": "VALIDATED",
    "last_updated": "2025-06-20",
    "next_review": "Continuous monitoring active",
    "certification": "Complete cure for death - 29 billion years validated"
  }
};

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

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

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

```
  const meterBar = document.getElementById('meterBar');
  const captureHint = document.getElementById('captureHint');

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

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

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

```
    const p = patterns[Math.floor(Math.random()*patterns.length)];
    return {timestamp:Date.now(), code:p.code, desc:p.desc,
phrases:p.phrases};
  }

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

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

  function mockModelResponse(decodedText, model){
    if(model === 'geto888g8678'){
      return `LLAMA- sim: ${decodedText.replace('    ','|')}\nConfidence: $
```

```
{Math.floor(Math.random()*15)+82}%`;
    } else {
      return `CLAUDE-sim: Interpreting the decoded pattern, this
suggests: ${decodedText.split('    ').pop().trim()}. Possible contexts:
memory, attention, or intent. Confidence ~ $
{Math.floor(Math.random()*12)+78}%`;
    }
  }

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

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

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

```
      const response = mockModelResponse(decoded === '— none —' ?
txt : decoded, model);
      pushMsg('AANIC', response);
    }
  };

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

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

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

```
    w.document.body.appendChild(pre);
    appendLog('Viewed DB: ' + key);
  };

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

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

  // Add placeholder DB (simple empty template)
  addPlaceholderBtn.onclick = ()=>{
```

```
    const key = 'PLACEHOLDER_' + Math.floor(Math.random()*10000);
    const template = {
      database_name: key,
      full_name: 'Placeholder DB ' + key,
      version: '0.0.1',
      created_by: 'User',
      creation_date: new Date().toISOString().slice(0,10),
      purpose: 'Placeholder / future DB',
      notes: 'This is a user-created placeholder DB'
    };
    DATABASES[key] = template;
    refreshDbList();
    dbSelect.value = key;
    appendLog('Added placeholder DB: ' + key);
    alert('Placeholder DB added: ' + key);
  };

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

 </script>
</body>
</html>
 s <!doctype html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>Adaptive Brain AI — Live API Integration</title>
<style>
 :root{--bg:#0f1724;--card:#111420;--accent:#00d1ff;--muted:#94a3b8}
 body{font-family:Inter,Segoe UI,Arial;background:linear-
gradient(180deg,#071023 0%, #071028
100%);color:#e6eef6;margin:0;padding:24px}
 .wrap{max-width:1000px;margin:0 auto;background:var(--
card);padding:20px;border-radius:12px;box-shadow:0 10px 30px
rgba(2,6,23,0.6)}
 header{display:flex;align-items:center;gap:16px}
 header h1{margin:0;color:var(--accent);font-size:20px}
 .controls{display:flex;gap:8px;margin-top:14px}
 input[type=text]{flex:1;padding:10px;border-radius:8px;border:1px
solid rgba(255,255,255,0.05);background:transparent;color:inherit}
 button{background:var(--
accent);color:#001;border:none;padding:10px 14px;border-
radius:8px;cursor:pointer;font-weight:600}
 .grid{display:grid;grid-template-columns:1fr 360px;gap:16px;margin-
```

```
top:18px}
  .panel{background:#0b1220;padding:12px;border-
radius:10px;border:1px solid rgba(255,255,255,0.02)}
  #output{height:420px;overflow:auto;padding:8px;font- family:ui-
monospace, SFMono- Regular, Menlo, Monaco, monospace;white-
space:pre- wrap}
  .tasks{display:flex;flex- wrap:wrap;gap:8px;margin- top:10px}
  .task{background:transparent;border:1px solid
rgba(255,255,255,0.04);padding:8px 10px;border-
radius:8px;cursor:pointer}
  .dac{margin- top:12px}
  canvas{width:100%;height:140px;border- radius:8px;background:linear-
gradient(180deg, rgba(0,0,0,0.15), rgba(255,255,255,0.02));display:block}
  .small{font- size:12px;color:var(- - muted)}
  footer{margin- top:12px;font- size:12px;color:var(- - muted)}
< /style>
< /head>
< body>
  < div class="wrap">
    < header>
      < h1> AugVT — Adaptive Brain AI (Live API Ready)< /h1>
      < div class="small"> AANIC DAC • 7 Forms • DB 82698 • LLaMA •
Claude< /div>
    < /header>

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

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

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

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

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

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

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

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

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

function drawWave(samples){
 ctx.clearRect(0,0,width,height);
 ctx.beginPath();
 const step = width / samples.length;
 for(let i=0;i<samples.length;i++){
  const x = i*step;
  const v = (samples[i]/255)*height;
  const y = height - v;
  if(i===0) ctx.moveTo(x,y); else ctx.lineTo(x,y);
```

```
  }
  ctx.strokeStyle = '#00d1ff'; ctx.lineWidth = 2; ctx.stroke();
}

function simulateDAC(input){
  // create pseudo-random waveform from input
  const bytes = input.split('').map(c=>c.charCodeAt(0)%256);
  const memFactor = sessionMemory.length % 32;
  const samples = new Array(200).fill(0).map((_,i)=>{
    const idx = i % Math.max(1,bytes.length);
    return (bytes.length?((bytes[idx]+memFactor*(i%3))%256):
((i*7+memFactor)%256));
  });
  drawWave(samples);
  dacHexEl.textContent =
samples.slice(0,8).map(n=>n.toString(16).padStart(2,'0')).join(' ');
  return samples;
}

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

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

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

```
  );
});

// Quick tasks
document.querySelectorAll('.task').forEach(btn=>
btn.addEventListener('click', ()=>{
  const task = btn.dataset.task; const val = inputEl.value.trim(); if(!val)
{ appendOutput('Type thoughts first.'); return; }
  sessionMemory.push(val);
memCountEl.textContent=sessionMemory.length; simulateDAC(val);
  let out='';
  if(task==='summarize') out='Summary: '+summarize(val);
  if(task==='translate') out='Translate (sim): '+translate(val);
  if(task==='emotion') out='Emotion: '+emotion(val);
  if(task==='seven') out='7- Forms:\n'+sevenForms(val);
  if(task==='db') callDB(val).then(r=> appendOutput('DB 82698 result:
\n'+JSON.stringify(r))).catch(e=> appendOutput('DB error: '+e.message));
  if(out) appendOutput(out);
}));

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

```
}

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

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

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

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

</script>
</body>
</html>
```