```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>NEUROMESH - Hedera Healthcare AI Network</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

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

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

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

        .header h1 {
            font-size: 2.5rem;
            margin-bottom: 10px;
            text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
```

```css
}

.header p {
    font-size: 1.2rem;
    opacity: 0.9;
}

.network-overview {
    background: rgba(255,255,255,0.95);
    border-radius: 15px;
    padding: 25px;
    margin-bottom: 30px;
    backdrop-filter: blur(10px);
    box-shadow: 0 8px 32px rgba(0,0,0,0.1);
}

.stats-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}

.stat-card {
    background: white;
    padding: 20px;
    border-radius: 10px;
    text-align: center;
    box-shadow: 0 4px 15px rgba(0,0,0,0.1);
    transition: transform 0.3s ease;
}

.stat-card:hover {
    transform: translateY(-5px);
}

.stat-value {
    font-size: 2rem;
    font-weight: bold;
    color: #667eea;
    margin-bottom: 5px;
}

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

.hospital-node {
    background: white;
    border-radius: 15px;
    padding: 20px;
    box-shadow: 0 8px 32px rgba(0,0,0,0.1);
    transition: all 0.3s ease;
}

.hospital-node:hover {
    transform: translateY(-3px);
    box-shadow: 0 12px 40px rgba(0,0,0,0.15);
}

.hospital-header {
    display: flex;
    align-items: center;
    margin-bottom: 15px;
}

.hospital-icon {
    width: 40px;
    height: 40px;
    background: linear-gradient(45deg, #667eea, #764ba2);
    border-radius: 8px;
    display: flex;
    align-items: center;
    justify-content: center;
    color: white;
    font-weight: bold;
    margin-right: 15px;
}

.hospital-name {
    font-size: 1.3rem;
    font-weight: bold;
    color: #333;
}

.training-status {
    display: flex;
    align-items: center;
    margin-bottom: 10px;
}

.status-indicator {
    width: 12px;
    height: 12px;
    border-radius: 50%;
    margin-right: 10px;
    animation: pulse 2s infinite;
}

.status-training { background: #ff6b6b; }
.status-ready { background: #4ecdc4; }
.status-uploading { background: #ffe66d; }

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

.model-metrics {
    background: #f8f9fa;
```

```css
    border-radius: 8px;
    padding: 15px;
    margin: 15px 0;
}

.metric-row {
    display: flex;
    justify-content: space-between;
    margin-bottom: 8px;
}

.progress-bar {
    width: 100%;
    height: 8px;
    background: #e9ecef;
    border-radius: 4px;
    overflow: hidden;
    margin: 10px 0;
}

.progress-fill {
    height: 100%;
    background: linear-gradient(90deg, #667eea, #764ba2);
    transition: width 0.3s ease;
}

.blockchain-section {
    background: rgba(255,255,255,0.95);
    border-radius: 15px;
    padding: 25px;
    margin-bottom: 30px;
}

.transaction-log {
    max-height: 300px;
    overflow-y: auto;
    background: #f8f9fa;
    border-radius: 8px;
    padding: 15px;
}

.transaction {
    background: white;
    border-left: 4px solid #667eea;
    padding: 10px 15px;
    margin-bottom: 10px;
    border-radius: 4px;
    font-family: monospace;
    font-size: 0.9rem;
}

.controls {
    display: flex;
    gap: 15px;
    justify-content: center;
    flex-wrap: wrap;
}

.btn {
    padding: 12px 25px;
    border: none;
    border-radius: 8px;
    font-size: 1rem;
    font-weight: bold;
    cursor: pointer;
    transition: all 0.3s ease;
    color: white;
}

.btn-primary {
    background: linear-gradient(45deg, #667eea, #764ba2);
}

.btn-secondary {
    background: linear-gradient(45deg, #4ecdc4, #44a08d);
}
```

```
.btn-danger {
    background: linear-gradient(45deg, #ff6b6b, #ee5a24);
}

.btn:hover {
    transform: translateY(-2px);
    box-shadow: 0 5px 15px rgba(0,0,0,0.2);
}

.btn:disabled {
    opacity: 0.6;
    cursor: not-allowed;
    transform: none;
}

.encryption-demo {
    background: #2c3e50;
    color: white;
    border-radius: 10px;
    padding: 15px;
    margin: 15px 0;
    font-family: monospace;
    font-size: 0.9rem;
}

.global-model-section {
    background: linear-gradient(135deg, #4ecdc4, #44a08d);
    color: white;
    border-radius: 15px;
    padding: 25px;
    text-align: center;
}

.global-accuracy {
    font-size: 3rem;
    font-weight: bold;
    margin: 20px 0;
```

```html
            text-shadow: 2px 2px 4px rgba(0,0,0,0.3);
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>🧠 NEUROMESH</h1>
            <p>Decentralized Neural Network Training on Hedera Blockchain</p>
        </div>

        <div class="network-overview">
            <h2>Network Overview</h2>
            <div class="stats-grid">
                <div class="stat-card">
                    <div class="stat-value" id="totalNodes">4</div>
                    <div>Active Nodes</div>
                </div>
                <div class="stat-card">
                    <div class="stat-value" id="totalTransactions">0</div>
                    <div>Blockchain Transactions</div>
                </div>
                <div class="stat-card">
                    <div class="stat-value" id="totalPatients">12,450</div>
                    <div>Total Patients (Private)</div>
                </div>
                <div class="stat-card">
                    <div class="stat-value" id="currentRound">0</div>
                    <div>Training Round</div>
                </div>
            </div>
        </div>

        <div class="hospital-grid" id="hospitalGrid">
            <!-- Hospital nodes will be generated here -->
        </div>

        <div class="blockchain-section">
```

```
            <h2>🔗 Hedera Consensus Service - Transaction Log</h2>
            <div class="transaction-log" id="transactionLog">
                <div class="transaction">
                    <strong>Genesis Block:</strong> NEUROMESH Network Initialized
                    <br><small>Timestamp: 2025-09-19T10:00:00Z | Topic: 0.0.1001</small>
                </div>
            </div>
        </div>

        <div class="global-model-section">
            <h2>🧠 Global Model Performance</h2>
            <div class="global-accuracy" id="globalAccuracy">85.2%</div>
            <p>Aggregated from all participating hospitals without exposing private data</p>
        </div>

        <div class="controls">
            <button class="btn btn-primary" onclick="startTrainingRound()">Start Training Round</
button>
            <button class="btn btn-secondary" onclick="aggregateModels()">Aggregate Models</
button>
            <button class="btn btn-danger" onclick="resetNetwork()">Reset Network</button>
        </div>
    </div>

    <script>
        // Hospital configuration
        const hospitals = [
            { id: 1, name: "St. Mary's Hospital", patients: 3200, accuracy: 84.5 },
            { id: 2, name: "General Medical Center", patients: 2800, accuracy: 86.1 },
            { id: 3, name: "University Hospital", patients: 4150, accuracy: 85.8 },
            { id: 4, name: "City Clinic Network", patients: 2300, accuracy: 83.9 }
        ];

        let currentRound = 0;
        let totalTransactions = 0;
        let isTraining = false;

        // Encryption simulation
        function simulateEncryption(data) {
            const chars =
'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789';
            return Array.from({length: 64}, () => chars.charAt(Math.floor(Math.random() *
chars.length))).join('');
        }

        // Initialize hospital nodes
        function initializeHospitals() {
            const grid = document.getElementById('hospitalGrid');
            grid.innerHTML = '';

            hospitals.forEach(hospital => {
                const nodeDiv = document.createElement('div');
                nodeDiv.className = 'hospital-node';
                nodeDiv.innerHTML = `
                    <div class="hospital-header">
                        <div class="hospital-icon">H${hospital.id}</div>
                        <div class="hospital-name">${hospital.name}</div>
                    </div>

                    <div class="training-status">
                        <div class="status-indicator status-ready" id="status-${hospital.id}"></div>
                        <span id="status-text-${hospital.id}">Ready for Training</span>
                    </div>

                    <div class="model-metrics">
                        <div class="metric-row">
                            <span>Local Dataset:</span>
                            <span><strong>${hospital.patients.toLocaleString()} patients</strong></span>
                        </div>
                        <div class="metric-row">
                            <span>Local Accuracy:</span>
                            <span><strong id="accuracy-${hospital.id}">${hospital.accuracy}%</strong></
span>
                        </div>
                        <div class="metric-row">
                            <span>Training Progress:</span>
                            <span><strong id="progress-${hospital.id}">0%</strong></span>
                        </div>
                        <div class="progress-bar">
                            <div class="progress-fill" id="progress-bar-${hospital.id}" style="width: 0%"></
div>
                        </div>
                    </div>

                    <div class="encryption-demo">
                        <strong>Encrypted Model Update:</strong>
                        <div id="encrypted-${hospital.id}" style="word-break: break-all; margin-top: 5px;
font-size: 0.8rem;">
                            Waiting for training...
                        </div>
                    </div>
                `;
                grid.appendChild(nodeDiv);
            });
        }

        // Add transaction to blockchain log
        function addTransaction(message, details = '') {
            totalTransactions++;
            document.getElementById('totalTransactions').textContent = totalTransactions;

            const log = document.getElementById('transactionLog');
            const transaction = document.createElement('div');
            transaction.className = 'transaction';
            transaction.innerHTML = `
                <strong>TX #${totalTransactions}:</strong> ${message}
                <br><small>Timestamp: ${new Date().toISOString()} | Topic: 0.0.1001 ${details}</small>
            `;
            log.insertBefore(transaction, log.firstChild);

            // Keep only last 10 transactions visible
            while (log.children.length > 10) {
                log.removeChild(log.lastChild);
            }
```

```
}

// Simulate training round
async function startTrainingRound() {
    if (isTraining) return;
    isTraining = true;
    currentRound++;

    document.getElementById('currentRound').textContent = currentRound;
    addTransaction(`Training Round ${currentRound} Started`, '| Gas: 0.001 ℏ');

    // Update all hospitals to training status
    hospitals.forEach(async (hospital, index) => {
        const statusIndicator = document.getElementById(`status-${hospital.id}`);
        const statusText = document.getElementById(`status-text-${hospital.id}`);
        const progressBar = document.getElementById(`progress-bar-${hospital.id}`);
        const progressText = document.getElementById(`progress-${hospital.id}`);

        statusIndicator.className = 'status-indicator status-training';
        statusText.textContent = 'Training in Progress...';

        // Simulate training progress
        for (let progress = 0; progress <= 100; progress += 10) {
            await new Promise(resolve => setTimeout(resolve, 200));
            progressBar.style.width = `${progress}%`;
            progressText.textContent = `${progress}%`;
        }

        // Training complete - generate encrypted update
        statusIndicator.className = 'status-indicator status-uploading';
        statusText.textContent = 'Uploading Encrypted Update...';

        const encryptedUpdate = simulateEncryption('model_weights');
        document.getElementById(`encrypted-${hospital.id}`).textContent = encryptedUpdate;

        // Simulate slight accuracy improvement
        hospital.accuracy += Math.random() * 0.8 + 0.2;
        document.getElementById(`accuracy-${hospital.id}`).textContent =
hospital.accuracy.toFixed(1) + '%';

                await new Promise(resolve => setTimeout(resolve, 1000));

                addTransaction(`Encrypted Model Update from ${hospital.name}`, `| Size: 2.4MB | Hash:
${encryptedUpdate.substring(0, 16)}...`);

                statusIndicator.className = 'status-indicator status-ready';
                statusText.textContent = 'Update Submitted';
            });

            setTimeout(() => {
                isTraining = false;
            }, 5000);
        }

        // Aggregate models using federated averaging
        async function aggregateModels() {
            if (isTraining) return;

            addTransaction('Starting Federated Aggregation', '| Consensus Algorithm: HCS');

            // Calculate weighted average accuracy
            let totalPatients = hospitals.reduce((sum, h) => sum + h.patients, 0);
            let weightedAccuracy = hospitals.reduce((sum, h) => sum + (h.accuracy * h.patients), 0) /
totalPatients;

            // Add some federated learning improvement
            weightedAccuracy += Math.random() * 1.2 + 0.3;

            // Animate global accuracy update
            const globalAccuracyElement = document.getElementById('globalAccuracy');
            let currentAccuracy = parseFloat(globalAccuracyElement.textContent);
            let targetAccuracy = weightedAccuracy;

            const animateAccuracy = () => {
                currentAccuracy += (targetAccuracy - currentAccuracy) * 0.1;
                globalAccuracyElement.textContent = currentAccuracy.toFixed(1) + '%';
                if (Math.abs(targetAccuracy - currentAccuracy) > 0.05) {
                    requestAnimationFrame(animateAccuracy);
                }
            };
            animateAccuracy();

            await new Promise(resolve => setTimeout(resolve, 2000));
            addTransaction('Global Model Updated', `| New Accuracy: ${targetAccuracy.toFixed(1)}% |
Consensus Reached`);

            // Reset all hospital progress
            hospitals.forEach(hospital => {
                document.getElementById(`progress-bar-${hospital.id}`).style.width = '0%';
                document.getElementById(`progress-${hospital.id}`).textContent = '0%';
                document.getElementById(`status-text-${hospital.id}`).textContent = 'Ready for Training';
            });
        }

        // Reset the network
        function resetNetwork() {
            currentRound = 0;
            totalTransactions = 0;
            isTraining = false;

            document.getElementById('currentRound').textContent = '0';
            document.getElementById('totalTransactions').textContent = '0';
            document.getElementById('globalAccuracy').textContent = '85.2%';

            // Reset hospital data
            hospitals.forEach(hospital => {
                hospital.accuracy = Math.random() * 3 + 83; // Reset to 83-86%
                document.getElementById(`accuracy-${hospital.id}`).textContent =
hospital.accuracy.toFixed(1) + '%';
                document.getElementById(`progress-bar-${hospital.id}`).style.width = '0%';
                document.getElementById(`progress-${hospital.id}`).textContent = '0%';
                document.getElementById(`status-text-${hospital.id}`).textContent = 'Ready for Training';
                document.getElementById(`encrypted-${hospital.id}`).textContent = 'Waiting for
training...';
            });

            // Clear transaction log
            const log = document.getElementById('transactionLog');
            log.innerHTML = `
                <div class="transaction">
                    <strong>Genesis Block:</strong> NEUROMESH Network Initialized
                    <br><small>Timestamp: 2025-09-19T10:00:00Z | Topic: 0.0.1001</small>
                </div>
            `;
        }

        // Initialize the demo
        document.addEventListener('DOMContentLoaded', function() {
            initializeHospitals();
        });
    </script>
</body>
</html>
```