```
import React, { useState, useMemo } from 'react';
import { LineChart, Line, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'rech

// ============================================================================
// DATA MODELS & TYPES
// ============================================================================

const generateId = () => Math.random().toString(36).substr(2, 9);

const NODE_TYPES = {
  DATACENTER: 'datacenter',
  EDGE_CLUSTER: 'edge_cluster',
  MIST_NODE: 'mist_node'
};

const REGION_COORDS = {
  'us-east-1': { lat: 40.7128, lng: -74.0060, name: 'New York' },
  'us-east-2': { lat: 40.7282, lng: -73.7949, name: 'Long Island' },
  'us-west-1': { lat: 37.7749, lng: -122.4194, name: 'San Francisco' },
  'us-west-2': { lat: 47.6062, lng: -122.3321, name: 'Seattle' },
  'eu-central-1': { lat: 50.1109, lng: 8.6821, name: 'Frankfurt' },
  'ap-south-1': { lat: 19.0760, lng: 72.8777, name: 'Mumbai' }
};

const calculateDistance = (lat1, lng1, lat2, lng2) => {
  const R = 6371;
  const dLat = (lat2 - lat1) * Math.PI / 180;
  const dLng = (lng2 - lng1) * Math.PI / 180;
  const a = Math.sin(dLat/2) * Math.sin(dLat/2) +
    Math.cos(lat1 * Math.PI / 180) * Math.cos(lat2 * Math.PI / 180) *
    Math.sin(dLng/2) * Math.sin(dLng/2);
  const c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
  return R * c;
};

// ============================================================================
// SEED DATA
// ============================================================================

const SEED_NODES = [
  {
    id: 'node-dc-1',
    type: NODE_TYPES.DATACENTER,
    name: 'North Shore DC',
    region: 'us-east-1',
    owner: 'North Shore Data',
    totalRAM: 2048,
    availableRAM: 1500,
    totalVRAM: 512,
    availableVRAM: 384,
    bandwidth: 10000,
    uptimeScore: 99.99,
    pricePerGBSec: 0.000002,
    lastSeen: Date.now()
  },
  {
    id: 'node-dc-2',
    type: NODE_TYPES.DATACENTER,
    name: 'EuroCore',
    region: 'eu-central-1',
    owner: 'EuroCore GmbH',
    totalRAM: 3072,
    availableRAM: 2200,
    totalVRAM: 768,
    availableVRAM: 600,
    bandwidth: 20000,
    uptimeScore: 99.95,
    pricePerGBSec: 0.0000018,
    lastSeen: Date.now()
  },
  {
    id: 'node-edge-1',
    type: NODE_TYPES.EDGE_CLUSTER,
    name: 'Brooklyn Gaming Hub',
    region: 'us-east-1',
    owner: 'NYC Esports',
    totalRAM: 384,
    availableRAM: 280,
    totalVRAM: 144,
    availableVRAM: 96,
    bandwidth: 2500,
    uptimeScore: 98.5,
    pricePerGBSec: 0.0000012,
    lastSeen: Date.now()
  },
  {
    id: 'node-edge-2',
    type: NODE_TYPES.EDGE_CLUSTER,
    name: 'Long Island Tech Lab',
    region: 'us-east-2',
    owner: 'LI University',
    totalRAM: 512,
    availableRAM: 420,
    totalVRAM: 192,
    availableVRAM: 156,
    bandwidth: 1000,
    uptimeScore: 97.2,
    pricePerGBSec: 0.0000010,
    lastSeen: Date.now()
  },
  {
    id: 'node-mist-1',
    type: NODE_TYPES.MIST_NODE,
    name: 'Alice_RTX4090',
    region: 'us-east-2',
    owner: 'alice_chen_42',
    totalRAM: 64,
    availableRAM: 48,
    totalVRAM: 24,
    availableVRAM: 18,
    bandwidth: 980,
    uptimeScore: 94.3,
    pricePerGBSec: 0.0000008,
    lastSeen: Date.now(),
    idleSchedule: '9am-5pm, 11pm-7am'
  },
  {
    id: 'node-mist-2',
    type: NODE_TYPES.MIST_NODE,
    name: 'Bob_Workstation',
    region: 'us-east-2',
    owner: 'bob_dev_studios',
    totalRAM: 128,
    availableRAM: 96,
    totalVRAM: 48,
    availableVRAM: 32,
    bandwidth: 850,
    uptimeScore: 91.7,
    pricePerGBSec: 0.0000007,
    lastSeen: Date.now(),
    idleSchedule: 'weekends, 7pm-9am'
  },
  {
    id: 'node-mist-3',
    type: NODE_TYPES.MIST_NODE,
    name: 'Carol_ML_Rig',
    region: 'us-east-1',
    owner: 'carol_mlops',
    totalRAM: 96,
    availableRAM: 72,
    totalVRAM: 40,
    availableVRAM: 28,
    bandwidth: 900,
    uptimeScore: 89.4,
    pricePerGBSec: 0.0000009,
    lastSeen: Date.now(),
    idleSchedule: 'nights only'
  },
  {
    id: 'node-mist-4',
    type: NODE_TYPES.MIST_NODE,
    name: 'David_Gaming_PC',
    region: 'us-west-1',
    owner: 'david_sf_gamer',
    totalRAM: 32,
    availableRAM: 24,
    totalVRAM: 16,
    availableVRAM: 12,
    bandwidth: 750,
    uptimeScore: 86.2,
    pricePerGBSec: 0.0000006,
    lastSeen: Date.now(),
    idleSchedule: 'workdays 9am-6pm'
  }
];

const SEED_CLIENTS = [
  { id: 'cli-1', name: 'Helix AI Labs', location: 'us-east-2', budget: 15000 },
  { id: 'cli-2', name: 'Orbital Analytics', location: 'us-west-1', budget: 30000
];

// ============================================================================
// MOCK API
// ============================================================================

class MockMAS {
  constructor() {
    this.nodes = [...SEED_NODES];
    this.clients = [...SEED_CLIENTS];
    this.contracts = [];

    setInterval(() => {
      this.nodes.forEach(node => {
        const ramDelta = (Math.random() - 0.5) * 20;
        const vramDelta = (Math.random() - 0.5) * 5;
        node.availableRAM = Math.max(0, Math.min(node.totalRAM, node.availableRAM
        node.availableVRAM = Math.max(0, Math.min(node.totalVRAM, node.availableV
        node.lastSeen = Date.now();
      });
    }, 5000);
  }

  getMarketplace(filters = {}) {
    let available = this.nodes.filter(n => n.availableRAM > 0 || n.availableVRAM

    if (filters.nodeType) {
      available = available.filter(n => n.type === filters.nodeType);
    }
    if (filters.region) {
      available = available.filter(n => n.region === filters.region);
    }
    if (filters.minRAM) {
      available = available.filter(n => n.availableRAM >= filters.minRAM);
    }
    if (filters.minVRAM) {
      available = available.filter(n => n.availableVRAM >= filters.minVRAM);
    }
    if (filters.minUptimeScore) {
      available = available.filter(n => n.uptimeScore >= filters.minUptimeScore);
    }

    return available;
  }

  requestMemory(clientId, requirements) {
    const client = this.clients.find(c => c.id === clientId);
    if (!client) return { error: 'Client not found' };

    const matches = this.matchNodes(client, requirements);

    return {
      clientId,
      requirements,
      matches: matches.slice(0, 5),
      timestamp: Date.now()
    };
  }

  matchNodes(client, requirements) {
    const { ramGB, vramGB, maxPricePerGBSec, preferLocal } = requirements;

    let candidates = this.nodes.filter(n =>
      n.availableRAM >= ramGB &&
      n.availableVRAM >= vramGB &&
      n.pricePerGBSec <= maxPricePerGBSec
    );

    const scored = candidates.map(node => {
      let score = 0;

      const clientCoords = REGION_COORDS[client.location];
      const nodeCoords = REGION_COORDS[node.region];
      if (clientCoords && nodeCoords) {
        const distance = calculateDistance(
          clientCoords.lat, clientCoords.lng,
          nodeCoords.lat, nodeCoords.lng
        );
        const proximityScore = Math.max(0, 100 - distance / 100);
        score += proximityScore * (preferLocal ? 3 : 1);
      }

      const priceScore = (1 - node.pricePerGBSec / maxPricePerGBSec) * 50;
      score += priceScore;

      const reliabilityScore = node.uptimeScore * 0.5;
      score += reliabilityScore;

      const capacityRatio = (node.availableRAM + node.availableVRAM) / (ramGB + v
      const capacityScore = Math.min(30, capacityRatio * 10);
      score += capacityScore;

      return { ...node, matchScore: score };
    });

    scored.sort((a, b) => b.matchScore - a.matchScore);
    return scored;
  }

  createContract(clientId, nodeId, ramGB, vramGB, durationSec) {
    const client = this.clients.find(c => c.id === clientId);
    const node = this.nodes.find(n => n.id === nodeId);

    if (!client || !node) return { error: 'Client or node not found' };
    if (node.availableRAM < ramGB || node.availableVRAM < vramGB) {
      return { error: 'Insufficient capacity' };
    }

    const contract = {
      id: generateId(),
      clientId,
      nodeId,
      ramGB,
      vramGB,
      durationSec,
      pricePerGBSec: node.pricePerGBSec,
      status: 'active',
      startTime: Date.now(),
      endTime: Date.now() + durationSec * 1000,
      totalCost: ((ramGB + vramGB) * durationSec * node.pricePerGBSec).toFixed(4)
    };

    node.availableRAM -= ramGB;
    node.availableVRAM -= vramGB;

    this.contracts.push(contract);
    return contract;
  }

  getClusterStats() {
    const clusters = {};

    this.nodes.forEach(node => {
      if (!clusters[node.region]) {
        clusters[node.region] = {
          region: node.region,
          coords: REGION_COORDS[node.region],
          nodeCount: 0,
          totalRAM: 0,
          availableRAM: 0,
          totalVRAM: 0,
          availableVRAM: 0,
          avgPrice: 0,
          nodeTypes: { datacenter: 0, edge_cluster: 0, mist_node: 0 }
        };
      }

      const cluster = clusters[node.region];
      cluster.nodeCount++;
      cluster.totalRAM += node.totalRAM;
      cluster.availableRAM += node.availableRAM;
      cluster.totalVRAM += node.totalVRAM;
      cluster.availableVRAM += node.availableVRAM;
      cluster.avgPrice += node.pricePerGBSec;
      cluster.nodeTypes[node.type]++;
    });
    Object.values(clusters).forEach(cluster => {
      cluster.avgPrice = cluster.avgPrice / cluster.nodeCount;
    });

    return Object.values(clusters);
  }

  getNodeEarnings(nodeId) {
    const contracts = this.contracts.filter(c => c.nodeId === nodeId);
    return contracts.reduce((sum, c) => sum + parseFloat(c.totalCost), 0);
  }
}

// ============================================================================
// MAIN APP
// ============================================================================

export default function MnemoV2() {
  const [api] = useState(() => new MockMAS());
  const [view, setView] = useState('buyer');
  const [showReadme, setShowReadme] = useState(false);

  return (
    <div style={{ fontFamily: 'system-ui, sans-serif', height: '100vh', display:
      <header style={{ background: 'linear-gradient(135deg, #667eea 0%, #764ba2 1
        <div style={{ display: 'flex', alignItems: 'center', gap: '1rem' }}>
          <h1 style={{ margin: 0, fontSize: '1.5rem', fontWeight: 700 }}>⚡ Mnemc
          <span style={{ opacity: 0.9, fontSize: '0.9rem' }}>VRAM & RAM Arbitrage
        </div>
        <div style={{ display: 'flex', gap: '1rem', alignItems: 'center' }}>
          <button
            onClick={() => setView('buyer')}
            style={{
              padding: '0.5rem 1rem',
              background: view === 'buyer' ? 'rgba(255,255,255,0.3)' : 'rgba(255,
              border: 'none',
              borderRadius: '6px',
              color: 'white',
              cursor: 'pointer',
              fontWeight: 500
            }}
          >
            Renter
          </button>
          <button
            onClick={() => setView('provider')}
            style={{
```

```
          padding: '0.5rem 1rem',
          background: view === 'provider' ? 'rgba(255,255,255,0.3)' : 'rgba(2
          border: 'none',
          borderRadius: '6px',
          color: 'white',
          cursor: 'pointer',
          fontWeight: 500
        }}
      >
        Provider
      </button>
      <button
        onClick={() => setView('clusters')}
        style={{
          padding: '0.5rem 1rem',
          background: view === 'clusters' ? 'rgba(255,255,255,0.3)' : 'rgba(2
          border: 'none',
          borderRadius: '6px',
          color: 'white',
          cursor: 'pointer',
          fontWeight: 500
        }}
      >
        Clusters
      </button>
      <button
        onClick={() => setShowReadme(!showReadme)}
        style={{
          padding: '0.5rem 1rem',
          background: 'rgba(255,255,255,0.2)',
          border: '1px solid rgba(255,255,255,0.3)',
          borderRadius: '6px',
          color: 'white',
          cursor: 'pointer',
          fontWeight: 500
        }}
      >
        README
      </button>
    </div>
  </header>

  <div style={{ flex: 1, overflow: 'auto', background: '#f8f9fa' }}>
    {showReadme ? (
      <ReadmePanel onClose={() => setShowReadme(false)} />
    ) : view === 'buyer' ? (
      <BuyerDashboard api={api} />
```

```
        ) : view === 'provider' ? (
          <ProviderDashboard api={api} />
        ) : (
          <ClusterView api={api} />
        )}
      </div>
    </div>
  );
}

// ============================================================================
// BUYER DASHBOARD
// ============================================================================

function BuyerDashboard({ api }) {
  const [filters, setFilters] = useState({});
  const [requirements, setRequirements] = useState({
    ramGB: 32,
    vramGB: 12,
    durationSec: 3600,
    maxPricePerGBSec: 0.000002,
    preferLocal: true
  });
  const [matchResults, setMatchResults] = useState(null);
  const [showRequestModal, setShowRequestModal] = useState(false);

  const client = api.clients[0];
  const marketplace = api.getMarketplace(filters);

  const handleRequestMemory = () => {
    const results = api.requestMemory(client.id, requirements);
    setMatchResults(results);
    setShowRequestModal(true);
  };

  const getNodeTypeLabel = (type) => {
    const labels = {
      datacenter: { label: 'Data Center', color: '#3b82f6', icon: '🏢' },
      edge_cluster: { label: 'Edge Cluster', color: '#10b981', icon: '🎮' },
      mist_node: { label: 'Mist Node', color: '#8b5cf6', icon: '💻' }
    };
    return labels[type] || labels.mist_node;
  };

  return (
    <div style={{ padding: '2rem' }}>
      <div style={{ background: 'white', padding: '1.5rem', borderRadius: '8px',
        <h2 style={{ margin: '0 0 0.5rem 0' }}>{client.name}</h2>
        <div style={{ color: '#6b7280' }}>Location: {REGION_COORDS[client.locatio
      </div>

      <div style={{ background: 'white', padding: '1.5rem', borderRadius: '8px',
        <h3 style={{ margin: '0 0 1rem 0' }}>Request Memory</h3>
        <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fit, min
          <div>
            <label style={{ display: 'block', fontSize: '0.85rem', color: '#6b728
            <input
              type="number"
              value={requirements.ramGB}
              onChange={e => setRequirements({ ...requirements, ramGB: parseInt(e
              style={{ width: '100%', padding: '0.5rem', border: '1px solid #d1d5
            />
          </div>
          <div>
            <label style={{ display: 'block', fontSize: '0.85rem', color: '#6b728
            <input
              type="number"
              value={requirements.vramGB}
              onChange={e => setRequirements({ ...requirements, vramGB: parseInt(
              style={{ width: '100%', padding: '0.5rem', border: '1px solid #d1d5
            />
          </div>
          <div>
            <label style={{ display: 'block', fontSize: '0.85rem', color: '#6b728
            <input
              type="number"
              value={requirements.durationSec / 3600}
              onChange={e => setRequirements({ ...requirements, durationSec: pars
              style={{ width: '100%', padding: '0.5rem', border: '1px solid #d1d5
            />
          </div>
        </div>
        <div style={{ marginBottom: '1rem' }}>
          <label style={{ display: 'flex', alignItems: 'center', gap: '0.5rem', c
            <input
              type="checkbox"
              checked={requirements.preferLocal}
              onChange={e => setRequirements({ ...requirements, preferLocal: e.ta
            />
            <span style={{ fontSize: '0.85rem' }}>Prefer local nodes (proximity-f
          </label>
        </div>
        <button
          onClick={handleRequestMemory}
          style={{ padding: '0.75rem 1.5rem', background: '#667eea', color: 'whit
        >
          Find Best Match
        </button>
      </div>

      <div style={{ background: 'white', padding: '1.5rem', borderRadius: '8px',
        <h3 style={{ margin: '0 0 1rem 0' }}>Browse Marketplace</h3>
        <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fit, min
          <div>
            <label style={{ display: 'block', fontSize: '0.85rem', color: '#6b728
            <select
              value={filters.nodeType || ''}
              onChange={e => setFilters({ ...filters, nodeType: e.target.value ||
              style={{ width: '100%', padding: '0.5rem', border: '1px solid #d1d5
            >
              <option value="">All Types</option>
              <option value="datacenter">Data Centers</option>
              <option value="edge_cluster">Edge Clusters</option>
              <option value="mist_node">Mist Nodes</option>
            </select>
          </div>
          <div>
            <label style={{ display: 'block', fontSize: '0.85rem', color: '#6b728
            <select
              value={filters.region || ''}
              onChange={e => setFilters({ ...filters, region: e.target.value || u
              style={{ width: '100%', padding: '0.5rem', border: '1px solid #d1d5
            >
              <option value="">All Regions</option>
              {Object.entries(REGION_COORDS).map(([key, val]) => (
                <option key={key} value={key}>{val.name}</option>
              ))}
            </select>
          </div>
        </div>
      </div>

      <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fill, minm
        {marketplace.map(node => {
          const typeInfo = getNodeTypeLabel(node.type);
          const estimatedCost = ((requirements.ramGB + requirements.vramGB) * req

          return (
            <div
              key={node.id}
              style={{
                background: 'white',
                padding: '1.5rem',
                borderRadius: '8px',
                boxShadow: '0 1px 3px rgba(0,0,0,0.1)',
                border: `2px solid ${typeInfo.color}20`
              }}
            >
              <div style={{ display: 'flex', justifyContent: 'space-between', ali
                <div>
                  <div style={{ display: 'flex', alignItems: 'center', gap: '0.5r
                    <span style={{ fontSize: '1.2rem' }}>{typeInfo.icon}</span>
                    <h4 style={{ margin: 0, fontSize: '1.1rem' }}>{node.name}</h4
                  </div>
                  <div style={{ fontSize: '0.85rem', color: '#6b7280' }}>
                    {REGION_COORDS[node.region]?.name}
                  </div>
                </div>
                <span style={{
                  padding: '0.25rem 0.75rem',
                  background: `${typeInfo.color}15`,
                  color: typeInfo.color,
                  border: `1px solid ${typeInfo.color}40`,
                  borderRadius: '12px',
                  fontSize: '0.75rem',
                  fontWeight: 600
                }}>
                  {typeInfo.label}
                </span>
              </div>

              <div style={{ display: 'grid', gridTemplateColumns: '1fr 1fr', gap:
                <div>
                  <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>RAM</div
                  <div style={{ fontSize: '1.1rem', fontWeight: 600 }}>
                    {node.availableRAM.toFixed(0)} GB
                  </div>
                  <div style={{ fontSize: '0.7rem', color: '#9ca3af' }}>of {node.
                </div>
                <div>
                  <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>VRAM</di
                  <div style={{ fontSize: '1.1rem', fontWeight: 600 }}>
                    {node.availableVRAM.toFixed(0)} GB
                  </div>
                  <div style={{ fontSize: '0.7rem', color: '#9ca3af' }}>of {node.
                </div>
              </div>
```

```
  <div style={{ display: 'flex', gap: '0.5rem', marginBottom: '0.75re
    <span style={{ padding: '0.25rem 0.5rem', background: '#f3f4f6',
      💰 ${node.pricePerGBSec.toFixed(7)}/GB·s
    </span>
    <span style={{ padding: '0.25rem 0.5rem', background: node.uptime
      ⏱️ {node.uptimeScore.toFixed(1)}%
    </span>
  </div>

  <div style={{ padding: '0.75rem', background: '#f9fafb', borderRadi
    <div style={{ fontSize: '0.75rem', color: '#6b7280', marginBottom
      Est. cost for your request:
    </div>
    <div style={{ fontSize: '1.25rem', fontWeight: 600, color: '#10b9
      ${estimatedCost}
    </div>
  </div>

  <button
    onClick={() => {
      const contract = api.createContract(
        client.id,
        node.id,
        requirements.ramGB,
        requirements.vramGB,
        requirements.durationSec
      );
      if (contract.error) {
        alert(contract.error);
      } else {
        alert('Contract created! Memory allocated.');
      }
    }}
    style={{
      width: '100%',
      padding: '0.5rem',
      background: '#667eea',
      color: 'white',
      border: 'none',
      borderRadius: '6px',
      cursor: 'pointer',
      fontWeight: 600
    }}
  >
    Rent Memory
  </button>
</div>
```

```
      );
    })}
  </div>

  {showRequestModal && matchResults && (
    <div style={{ position: 'fixed', inset: 0, background: 'rgba(0,0,0,0.5)',
      <div style={{ background: 'white', padding: '2rem', borderRadius: '12px
        <h3 style={{ margin: '0 0 1rem 0' }}>Best Matches Found</h3>

        {matchResults.matches.map((node, idx) => {
          const typeInfo = getNodeTypeLabel(node.type);
          const cost = ((requirements.ramGB + requirements.vramGB) * requirem

          return (
            <div
              key={node.id}
              style={{
                padding: '1rem',
                marginBottom: '1rem',
                border: `2px solid ${typeInfo.color}30`,
                borderRadius: '8px',
                background: idx === 0 ? `${typeInfo.color}05` : 'white'
              }}
            >
              <div style={{ display: 'flex', justifyContent: 'space-between',
                <div style={{ flex: 1 }}>
                  <div style={{ fontWeight: 600, marginBottom: '0.25rem' }}>
                    {idx === 0 ? '🏆 ' : ''}{node.name}
                  </div>
                  <div style={{ fontSize: '0.85rem', color: '#6b7280' }}>
                    Score: {node.matchScore.toFixed(1)}
                  </div>
                </div>
                <div style={{ textAlign: 'right' }}>
                  <div style={{ fontSize: '1.25rem', fontWeight: 600, color:
                </div>
              </div>

              <button
                onClick={() => {
                  api.createContract(client.id, node.id, requirements.ramGB,
                  alert('Contract created with ' + node.name);
                  setShowRequestModal(false);
                }}
                style={{
                  width: '100%',
                  padding: '0.5rem',
                      background: idx === 0 ? '#667eea' : '#e5e7eb',
                      color: idx === 0 ? 'white' : '#374151',
                      border: 'none',
                      borderRadius: '6px',
                      cursor: 'pointer',
                      fontWeight: 500
                    }}
                  >
                    {idx === 0 ? 'Select Best Match' : 'Select This Node'}
                  </button>
                </div>
              );
            })}

            <button
              onClick={() => setShowRequestModal(false)}
              style={{ width: '100%', padding: '0.5rem', background: '#e5e7eb', b
            >
              Close
            </button>
          </div>
        </div>
      )}
    </div>
  );
}

// ===============================================================================
// PROVIDER DASHBOARD
// ===============================================================================

function ProviderDashboard({ api }) {
  const nodes = api.nodes;

  const getNodeTypeLabel = (type) => {
    const labels = {
      datacenter: { label: 'Data Center', color: '#3b82f6', icon: '🏢' },
      edge_cluster: { label: 'Edge Cluster', color: '#10b981', icon: '🎮' },
      mist_node: { label: 'Mist Node', color: '#8b5cf6', icon: '💻' }
    };
    return labels[type] || labels.mist_node;
  };

  return (
    <div style={{ padding: '2rem' }}>
      <h2 style={{ marginBottom: '1.5rem' }}>Provider Dashboard</h2>
```

```
        <div style={{ display: 'grid', gap: '1.5rem' }}>
          {nodes.map(node => {
            const typeInfo = getNodeTypeLabel(node.type);
            const earnings = api.getNodeEarnings(node.id);
            const utilization = ((node.totalRAM - node.availableRAM) / node.totalRA

            return (
              <div key={node.id} style={{ background: 'white', padding: '1.5rem', b
                <div style={{ display: 'flex', justifyContent: 'space-between', ali
                  <div>
                    <div style={{ display: 'flex', alignItems: 'center', gap: '0.5r
                      <span style={{ fontSize: '1.2rem' }}>{typeInfo.icon}</span>
                      <h3 style={{ margin: 0 }}>{node.name}</h3>
                    </div>
                    <div style={{ fontSize: '0.85rem', color: '#6b7280' }}>
                      {typeInfo.label} • {REGION_COORDS[node.region]?.name}
                    </div>
                    {node.idleSchedule && (
                      <div style={{ fontSize: '0.75rem', color: '#6b7280', marginTo
                        Idle: {node.idleSchedule}
                      </div>
                    )}
                  </div>
                  <div style={{ textAlign: 'right' }}>
                    <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>Earned</
                    <div style={{ fontSize: '1.5rem', fontWeight: 600, color: '#10b
                      ${earnings.toFixed(2)}
                    </div>
                  </div>
                </div>

                <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fi
                  <div>
                    <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>RAM</div
                    <div style={{ fontSize: '1.1rem', fontWeight: 600 }}>{node.tota
                  </div>
                  <div>
                    <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>VRAM</di
                    <div style={{ fontSize: '1.1rem', fontWeight: 600 }}>{node.tota
                  </div>
                  <div>
                    <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>Utilizat
                    <div style={{ fontSize: '1.1rem', fontWeight: 600 }}>{utilizati
                  </div>
                </div>

                <div style={{ marginBottom: '0.5rem', background: '#e5e7eb', height
              <div style={{ background: typeInfo.color, height: '100%', width:
            </div>
          </div>
        );
      })}
    </div>
  </div>
);
}

// ==============================================================================
// CLUSTER VIEW
// ==============================================================================

function ClusterView({ api }) {
  const clusters = api.getClusterStats();

  return (
    <div style={{ padding: '2rem' }}>
      <h2 style={{ marginBottom: '1rem' }}>Geographic Clusters</h2>
      <p style={{ color: '#6b7280', marginBottom: '2rem' }}>
        Local memory meshes organized by region. Proximity equals lower latency a
      </p>

      <div style={{ display: 'grid', gridTemplateColumns: 'repeat(auto-fill, minm
        {clusters.map(cluster => {
          const totalNodes = cluster.nodeCount;
          const mistNodePct = (cluster.nodeTypes.mist_node / totalNodes * 100).to

          return (
            <div key={cluster.region} style={{ background: 'white', padding: '1.5
              <h3 style={{ margin: '0 0 0.5rem 0' }}>
                📍 {cluster.coords?.name || cluster.region}
              </h3>
              <div style={{ fontSize: '0.85rem', color: '#6b7280', marginBottom:
                {cluster.nodeCount} nodes • {mistNodePct}% community-powered
              </div>

              <div style={{ display: 'grid', gridTemplateColumns: '1fr 1fr', gap:
                <div>
                  <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>Total RA
                  <div style={{ fontSize: '1.25rem', fontWeight: 600 }}>{cluster.
                </div>
                <div>
                  <div style={{ fontSize: '0.75rem', color: '#6b7280' }}>Total VR
                  <div style={{ fontSize: '1.25rem', fontWeight: 600 }}>{cluster.
                </div>
              </div>

              <div style={{ marginBottom: '1rem' }}>
                <div style={{ fontSize: '0.75rem', color: '#6b7280', marginBottom
                <div style={{ display: 'flex', gap: '0.5rem', flexWrap: 'wrap' }}
                  {cluster.nodeTypes.datacenter > 0 && (
                    <span style={{ padding: '0.25rem 0.5rem', background: '#3b82f
                      🏢 {cluster.nodeTypes.datacenter}
                    </span>
                  )}
                  {cluster.nodeTypes.edge_cluster > 0 && (
                    <span style={{ padding: '0.25rem 0.5rem', background: '#10b98
                      🎮 {cluster.nodeTypes.edge_cluster}
                    </span>
                  )}
                  {cluster.nodeTypes.mist_node > 0 && (
                    <span style={{ padding: '0.25rem 0.5rem', background: '#8b5cf
                      💻 {cluster.nodeTypes.mist_node}
                    </span>
                  )}
                </div>
              </div>
            </div>
          );
        })}
      </div>
    </div>
  );
}

// ===============================================================================
// README PANEL
// ===============================================================================

function ReadmePanel({ onClose }) {
  return (
    <div style={{ padding: '2rem', maxWidth: '900px', margin: '0 auto' }}>
      <div style={{ display: 'flex', justifyContent: 'space-between', alignItems:
        <h2 style={{ margin: 0 }}>📖 Mnemo - Memory Arbitrage System</h2>
        <button onClick={onClose} style={{ padding: '0.5rem 1rem', background: '#
          Close
        </button>
      </div>

      <div style={{ background: 'white', padding: '2rem', borderRadius: '8px', li
        <h3>What is Memory Arbitrage?</h3>
        <p>
```

```
          Mnemo captures idle VRAM and RAM from GPUs and consumer PCs, then rents
          While everyone rents whole GPUs, we rent the memory layer itself.
        </p>

        <h3>Three-Tier Architecture</h3>
        <ul>
          <li><strong>Data Centers:</strong> Enterprise providers with 99.9%+ upt
          <li><strong>Edge Clusters:</strong> Gaming cafes, university labs pooli
          <li><strong>Mist Nodes:</strong> Individual consumer machines earning p
        </ul>

        <h3>Local Network Vision</h3>
        <p>
          Geographic clusters outperform AWS/GCP through proximity. Long Island c
          achieves less than 1ms latency versus 10-30ms to AWS.
        </p>

        <h3>Economics</h3>
        <p>
          RTX 4090 with 24GB VRAM used 12GB for 4 hours/day leaves 240 GB-hours/d
          Scale to 1,000 machines creates instant 12TB memory pool.
        </p>

        <h3>How to Use</h3>
        <p><strong>As Renter:</strong> Enter RAM/VRAM needs, enable "prefer local
        <p><strong>As Provider:</strong> Install node agent, set idle schedule, e
      </div>
    </div>
  );
}
```