```
<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

    <meta name="viewport" content="width=device-width,
initial-scale=1.0">

    <title>Network Traffic Analyzer - Dashboard</title>

    <style>

        * {

            margin: 0;

            padding: 0;

            box-sizing: border-box;

        }


        body {

            font-family: 'Segoe UI', Tahoma, Geneva, Verdana,
sans-serif;

            background: linear-gradient(135deg, #1e3c72 0%, #2a5298
100%);

            color: #fff;

            min-height: 100vh;

        }


        .container {

            max-width: 1400px;

            margin: 0 auto;

            padding: 20px;

        }
```

```
        .header {
            text-align: center;
            margin-bottom: 30px;
            padding: 20px 0;
        }

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

        .header p {
            font-size: 1.1rem;
            opacity: 0.9;
        }

        .status-bar {
            display: flex;
            justify-content: space-between;
            align-items: center;
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            border-radius: 15px;
            padding: 15px 25px;
            margin-bottom: 30px;
```

```
            border: 1px solid rgba(255, 255, 255, 0.2);
        }

        .status-item {
            text-align: center;
        }

        .status-item .value {
            font-size: 1.5rem;
            font-weight: bold;
            color: #4ade80;
        }

        .status-item .label {
            font-size: 0.9rem;
            opacity: 0.8;
            margin-top: 5px;
        }

        .dashboard-grid {
            display: grid;
            grid-template-columns: repeat(auto-fit, minmax(400px,
1fr));
            gap: 20px;
            margin-bottom: 30px;
        }

        .card {
```

```
            background: rgba(255, 255, 255, 0.1);
            backdrop-filter: blur(10px);
            border-radius: 15px;
            padding: 25px;
            border: 1px solid rgba(255, 255, 255, 0.2);
            transition: transform 0.3s ease, box-shadow 0.3s ease;
        }

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

        .card h3 {
            font-size: 1.3rem;
            margin-bottom: 15px;
            color: #fbbf24;
        }

        .chart-container {
            height: 300px;
            position: relative;
        }

        .alert-item {
            background: rgba(239, 68, 68, 0.2);
            border-left: 4px solid #ef4444;
```

```
            padding: 15px;
            margin-bottom: 10px;
            border-radius: 8px;
            transition: background 0.3s ease;
        }

        .alert-item:hover {
            background: rgba(239, 68, 68, 0.3);
        }

        .alert-item.severity-1 { border-left-color: #10b981; }
        .alert-item.severity-2 { border-left-color: #f59e0b; }
        .alert-item.severity-3 { border-left-color: #ef4444; }
        .alert-item.severity-4 { border-left-color: #dc2626; }

        .alert-time {
            font-size: 0.8rem;
            opacity: 0.7;
        }

        .alert-type {
            font-weight: bold;
            color: #fbbf24;
        }

        .controls {
            display: flex;
```

```
            gap: 15px;
            margin-bottom: 20px;
            flex-wrap: wrap;
            align-items: center;
        }

        .control-group {
            display: flex;
            align-items: center;
            gap: 10px;
        }

        label {
            font-weight: 500;
        }

        input, select, button {
            padding: 8px 12px;
            border: 1px solid rgba(255, 255, 255, 0.3);
            border-radius: 8px;
            background: rgba(255, 255, 255, 0.1);
            color: #fff;
            font-size: 0.9rem;
        }

        input::placeholder {
            color: rgba(255, 255, 255, 0.6);
```

```
        }

        button {
            background: linear-gradient(135deg, #3b82f6, #1d4ed8);
            cursor: pointer;
            transition: all 0.3s ease;
            border: none;
            font-weight: 500;
        }

        button:hover {
            background: linear-gradient(135deg, #2563eb, #1e40af);
            transform: translateY(-2px);
            box-shadow: 0 5px 15px rgba(59, 130, 246, 0.3);
        }

        .refresh-btn {
            background: linear-gradient(135deg, #10b981, #059669);
        }

        .refresh-btn:hover {
            background: linear-gradient(135deg, #059669, #047857);
        }

        .table-container {
            overflow-x: auto;
            background: rgba(255, 255, 255, 0.05);
```

```
            border-radius: 10px;
            max-height: 400px;
        }

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

        th, td {
            padding: 12px;
            text-align: left;
            border-bottom: 1px solid rgba(255, 255, 255, 0.1);
        }

        th {
            background: rgba(255, 255, 255, 0.1);
            font-weight: 600;
            position: sticky;
            top: 0;
        }

        tr:hover {
            background: rgba(255, 255, 255, 0.05);
        }

        .loading {
```

```
            display: none;
            text-align: center;
            padding: 20px;
        }

        .spinner {
            border: 3px solid rgba(255, 255, 255, 0.3);
            border-radius: 50%;
            border-top: 3px solid #fff;
            width: 30px;
            height: 30px;
            animation: spin 1s linear infinite;
            margin: 0 auto 10px;
        }

        @keyframes spin {
            0% { transform: rotate(0deg); }
            100% { transform: rotate(360deg); }
        }

        .error {
            background: rgba(239, 68, 68, 0.2);
            border: 1px solid #ef4444;
            border-radius: 8px;
            padding: 15px;
            margin: 10px 0;
            display: none;
```

```
        }

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

            .status-bar {
                flex-direction: column;
                gap: 15px;
            }

            .controls {
                flex-direction: column;
                align-items: stretch;
            }
        }
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>🛡️ Network Traffic Analyzer</h1>
            <p>Real-time Network Monitoring & Security Analysis Dashboard</p>
        </div>

        <div class="status-bar">
```

```
            <div class="status-item">
                <div class="value" id="total-packets">0</div>
                <div class="label">Total Packets</div>
            </div>
            <div class="status-item">
                <div class="value" id="active-alerts">0</div>
                <div class="label">Active Alerts</div>
            </div>
            <div class="status-item">
                <div class="value" id="nodes-count">0</div>
                <div class="label">Active Nodes</div>
            </div>
            <div class="status-item">
                <div class="value" id="last-update">Never</div>
                <div class="label">Last Update</div>
            </div>
        </div>

        <div class="controls">
            <div class="control-group">
                <label for="time-range">Time Range:</label>
                <select id="time-range">
                    <option value="1">Last Hour</option>
                    <option value="6">Last 6 Hours</option>
                    <option value="24" selected>Last 24 Hours</option>
                    <option value="168">Last Week</option>
                </select>
```

```
            </div>
            <div class="control-group">
                <label for="auto-refresh">Auto Refresh:</label>
                <select id="auto-refresh">
                    <option value="0">Off</option>
                    <option value="30" selected>30 seconds</option>
                    <option value="60">1 minute</option>
                    <option value="300">5 minutes</option>
                </select>
            </div>
            <button class="refresh-btn" onclick="refreshDashboard()">🔄 Refresh Now</button>
        </div>

        <div class="loading" id="loading">
            <div class="spinner"></div>
            <div>Loading dashboard data...</div>
        </div>

        <div class="error" id="error">
            <strong>Error:</strong> <span id="error-message"></span>
        </div>

        <div class="dashboard-grid">
            <div class="card">
                <h3>📊 Traffic Statistics</h3>
                <div class="chart-container">
                    <canvas id="traffic-chart"></canvas>
```

```
                </div>
            </div>

            <div class="card">
                <h3>🚨 Recent Alerts</h3>
                <div id="recent-alerts" style="max-height: 300px; overflow-y: auto;">
                    <div class="alert-item">
                        <div class="alert-type">No alerts available</div>
                        <div class="alert-time">System initialized</div>
                    </div>
                </div>
            </div>

            <div class="card">
                <h3>🌐 Top Source IPs</h3>
                <div class="table-container">
                    <table>
                        <thead>
                            <tr>
                                <th>IP Address</th>
                                <th>Packets</th>
                                <th>Activity</th>
                            </tr>
                        </thead>
                        <tbody id="top-ips">
                            <tr>
```

```
                                <td colspan="3" style="text-align:
center;">Loading...</td>

                            </tr>

                        </tbody>

                    </table>

                </div>

            </div>


            <div class="card">

                <h3>🔍 Protocol Distribution</h3>

                <div class="chart-container">

                    <canvas id="protocol-chart"></canvas>

                </div>

            </div>

        </div>


        <div class="card">

            <h3>📋 Recent Packets</h3>

            <div class="controls" style="margin-bottom: 15px;">

                <div class="control-group">

                    <label for="filter-ip">Filter by IP:</label>

                    <input type="text" id="filter-ip"
placeholder="e.g., 192.168.1.100">

                </div>

                <div class="control-group">

                    <label for="filter-protocol">Protocol:</label>

                    <select id="filter-protocol">

                        <option value="">All Protocols</option>
```

```
                        <option value="TCP">TCP</option>
                        <option value="UDP">UDP</option>
                        <option value="ICMP">ICMP</option>
                        <option value="ARP">ARP</option>
                    </select>
                </div>
                <button onclick="filterPackets()">🔍 Apply
Filter</button>
            </div>

            <div class="table-container">
                <table>
                    <thead>
                        <tr>
                            <th>Timestamp</th>
                            <th>Source IP</th>
                            <th>Destination IP</th>
                            <th>Protocol</th>
                            <th>Ports</th>
                            <th>Size</th>
                        </tr>
                    </thead>
                    <tbody id="recent-packets">
                        <tr>
                            <td colspan="6" style="text-align:
center;">Loading...</td>
                        </tr>
                    </tbody>
```

```
                </table>
            </div>
        </div>
    </div>

    <script src="https://cdnjs.cloudflare.com/ajax/libs/Chart.js/3.9.1/chart.min.js"></script>
    <script>
        // Global variables
        const API_BASE = 'http://localhost:5000/api';
        let trafficChart, protocolChart;
        let autoRefreshInterval;

        // Initialize dashboard
        document.addEventListener('DOMContentLoaded', function() {
            initializeCharts();
            refreshDashboard();
            setupAutoRefresh();

            // Setup event listeners
            document.getElementById('time-range').addEventListener('change', refreshDashboard);
            document.getElementById('auto-refresh').addEventListener('change', setupAutoRefresh);
        });

        function initializeCharts() {
```

```
            // Traffic Statistics Chart
            const trafficCtx = document.getElementById('traffic-chart').getContext('2d');
            trafficChart = new Chart(trafficCtx, {
                type: 'doughnut',
                data: {
                    labels: ['TCP', 'UDP', 'ICMP', 'ARP', 'Other'],
                    datasets: [{
                        data: [0, 0, 0, 0, 0],
                        backgroundColor: [
                            '#3b82f6',
                            '#10b981',
                            '#f59e0b',
                            '#ef4444',
                            '#8b5cf6'
                        ],
                        borderWidth: 0
                    }]
                },
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                    plugins: {
                        legend: {
                            position: 'bottom',
                            labels: { color: '#fff' }
                        }
                    }
```

```
                }
            });

            // Protocol Distribution Chart
            const protocolCtx = document.getElementById('protocol-chart').getContext('2d');
            protocolChart = new Chart(protocolCtx, {
                type: 'bar',
                data: {
                    labels: [],
                    datasets: [{
                        label: 'Packet Count',
                        data: [],
                        backgroundColor: 'rgba(59, 130, 246, 0.6)',
                        borderColor: '#3b82f6',
                        borderWidth: 1
                    }]
                },
                options: {
                    responsive: true,
                    maintainAspectRatio: false,
                    plugins: {
                        legend: {
                            labels: { color: '#fff' }
                        }
                    },
                    scales: {
                        x: {
```

```
                                ticks: { color: '#fff' },
                                grid: { color: 'rgba(255, 255, 255, 0.1)' }
                            },
                            y: {
                                ticks: { color: '#fff' },
                                grid: { color: 'rgba(255, 255, 255, 0.1)' }
                            }
                        }
                    }
                });
            }

            function showLoading(show = true) {
                document.getElementById('loading').style.display = show ? 'block' : 'none';
            }

            function showError(message) {
                document.getElementById('error-message').textContent = message;
                document.getElementById('error').style.display = 'block';
            }

            function hideError() {
                document.getElementById('error').style.display = 'none';
            }

            async function fetchData(endpoint, params = {}) {
```

```
            try {
                const url = new URL(`${API_BASE}${endpoint}`);
                Object.keys(params).forEach(key =>
url.searchParams.append(key, params[key]));

                const response = await fetch(url);
                if (!response.ok) {
                    throw new Error(`HTTP error! status:
${response.status}`);
                }

                return await response.json();
            } catch (error) {
                console.error('API Error:', error);
                throw error;
            }
        }

        async function refreshDashboard() {
            showLoading(true);
            hideError();

            try {
                const timeRange =
document.getElementById('time-range').value;

                // Fetch all data in parallel
                const [stats, alerts, packets, nodes] = await
Promise.all([
```

```
                    fetchData('/stats', { hours: timeRange }),
                    fetchData('/alerts', { limit: 10, resolved: false }),
                    fetchData('/packets', { limit: 50 }),
                    fetchData('/nodes')
                ]);

                // Update status bar
                updateStatusBar(stats, alerts, nodes);

                // Update charts
                updateCharts(stats);

                // Update alerts
                updateAlerts(alerts.alerts);

                // Update top IPs
                updateTopIPs(stats.top_source_ips);

                // Update recent packets
                updateRecentPackets(packets.packets);

                // Update last update time
                document.getElementById('last-update').textContent = new Date().toLocaleTimeString();

            } catch (error) {
                showError(`Failed to refresh dashboard: ${error.message}`);
```

```
            } finally {
                showLoading(false);
            }
        }

        function updateStatusBar(stats, alerts, nodes) {
            // Calculate total packets
            const totalPackets = stats.traffic_stats.reduce((sum, stat) => sum + stat.packet_count, 0);
            document.getElementById('total-packets').textContent = totalPackets.toLocaleString();

            // Active alerts
            document.getElementById('active-alerts').textContent = alerts.count;

            // Active nodes
            document.getElementById('nodes-count').textContent = nodes.count;
        }

        function updateCharts(stats) {
            // Update traffic chart (doughnut)
            const protocolData = stats.traffic_stats;
            const labels = protocolData.map(item => item.protocol);
            const data = protocolData.map(item => item.packet_count);

            trafficChart.data.labels = labels;
            trafficChart.data.datasets[0].data = data;
```

```
            trafficChart.update();


            // Update protocol chart (bar)
            protocolChart.data.labels = labels;
            protocolChart.data.datasets[0].data = data;
            protocolChart.update();
        }

        function updateAlerts(alerts) {
            const container = document.getElementById('recent-alerts');

            if (alerts.length === 0) {
                container.innerHTML = '<div class="alert-item"><div
class="alert-type">No active alerts</div><div class="alert-time">All
clear</div></div>';
                return;
            }

            container.innerHTML = alerts.map(alert => {
                const time = new
Date(alert.timestamp).toLocaleString();
                return `
                    <div class="alert-item severity-${alert.severity}">
                        <div
class="alert-type">${alert.alert_type}</div>
                        <div>Source: ${alert.source_ip}</div>
                        <div>${alert.description}</div>
                        <div class="alert-time">${time}</div>
                    </div>
```

```
                `;
            }).join('');
        }

        function updateTopIPs(topIPs) {
            const tbody = document.getElementById('top-ips');

            if (topIPs.length === 0) {
                tbody.innerHTML = '<tr><td colspan="3"
style="text-align: center;">No data available</td></tr>';
                return;
            }

            tbody.innerHTML = topIPs.map(ip => {
                const activity = ip.packet_count > 1000 ? 'High' :
ip.packet_count > 100 ? 'Medium' : 'Low';
                return `
                    <tr>
                        <td>${ip.source_ip}</td>
                        <td>${ip.packet_count.toLocaleString()}</td>
                        <td><span style="color: ${activity === 'High' ?
'#ef4444' : activity === 'Medium' ? '#f59e0b' :
'#10b981'}">${activity}</span></td>
                    </tr>
                `;
            }).join('');
        }

        function updateRecentPackets(packets) {
```

```
            const tbody = document.getElementById('recent-packets');


            if (packets.length === 0) {
                tbody.innerHTML = '<tr><td colspan="6"
style="text-align: center;">No packets found</td></tr>';
                return;
            }


            tbody.innerHTML = packets.map(packet => {
                const time = new
Date(packet.timestamp).toLocaleString();
                const ports = `${packet.source_port ||
'N/A'}:${packet.destination_port || 'N/A'}`;
                return `
                    <tr>
                        <td>${time}</td>
                        <td>${packet.source_ip}</td>
                        <td>${packet.destination_ip}</td>
                        <td><span style="color:
#fbbf24">${packet.protocol}</span></td>
                        <td>${ports}</td>
                        <td>${packet.packet_size} bytes</td>
                    </tr>
                `;
            }).join('');
        }


        async function filterPackets() {
            showLoading(true);
```

```
            try {
                const ip = document.getElementById('filter-ip').value;
                const protocol =
document.getElementById('filter-protocol').value;

                const params = { limit: 50 };
                if (ip) params.source_ip = ip;
                if (protocol) params.protocol = protocol;

                const data = await fetchData('/packets', params);
                updateRecentPackets(data.packets);

            } catch (error) {
                showError(`Failed to filter packets:
${error.message}`);
            } finally {
                showLoading(false);
            }
        }

        function setupAutoRefresh() {
            if (autoRefreshInterval) {
                clearInterval(autoRefreshInterval);
            }

            const interval =
parseInt(document.getElementById('auto-refresh').value);
            if (interval > 0) {
```

```
                autoRefreshInterval = setInterval(refreshDashboard,
interval * 1000);

            }

        }


        // Utility functions

        function formatBytes(bytes) {

            if (bytes === 0) return '0 B';

            const k = 1024;

            const sizes = ['B', 'KB', 'MB', 'GB'];

            const i = Math.floor(Math.log(bytes) / Math.log(k));

            return parseFloat((bytes / Math.pow(k, i)).toFixed(2)) + '
' + sizes[i];

        }


        function formatTime(timestamp) {

            return new Date(timestamp).toLocaleString();

        }


        // Handle connection errors gracefully

        window.addEventListener('online', function() {

            hideError();

            refreshDashboard();

        });


        window.addEventListener('offline', function() {

            showError('Connection lost. Dashboard will update when
connection is restored.');

        });
```

```
    </script>

</body>

</html>
```