LOVELY
PROFESSIONAL
UNIVERSITY

Name – Saurav Suman
Section – K23HG
Roll No - 49

Real-Time Memory Allocation Tracker

Project Overview

- - A web-based real-time memory tracking visualization tool.
- - Offers insights for developers and system administrators.
- - Interactive UI, live visualization, and history tracking features.

Module-Wise Breakdown

- 1. UI Module (HTML, CSS)

- Accessible and responsive design.

- 2. Memory Tracking (JavaScript)

- Tracks deallocation and allocation.

- 3. Visualization (Chart.js)

- Offers realtime graph updation.

- 4. Data Simulation (MemorySimulator)

- Simulates memory activities.

- 5. User Controls - Includes Start, Stop, and Clear Data features.

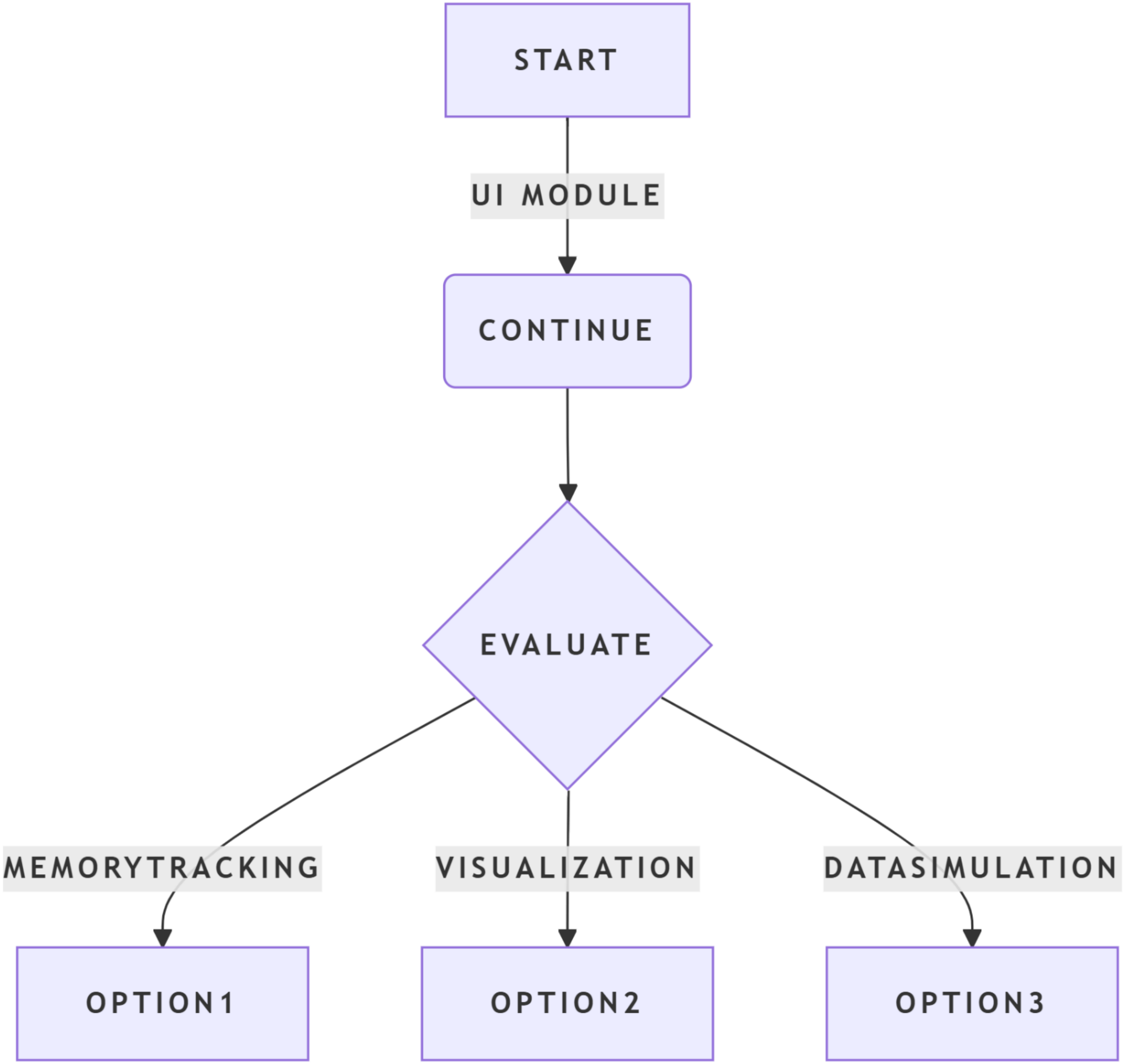
Key Functionalities

- Start/Stop Memory Tracking • Real-time Graph Updates

- View Metrics:
- Total Memory • Used Memory • Free Memory • Maintain Allocation History • Responsive Web Design

Technology Used

• - HTML, CSS, JavaScript • - Chart.js for Visualization .


START
UI MODULE
CONTINUE
EVALUATE
MEMORYTRACKING
VISUALIZATION
DATASIMULATION
OPTION1
OPTION2
OPTION3

Revision Tracking on GitHub

- - Repository: **Real-Time-Memory-Allocation-Tracker**
- https://github.com/sauravsuman18/Real-Time-Memory-Allocation-Tracker)

Problem Statement

•Efficient management of memory allocation is paramount in software development.

- Problems:
  Performance bottleneck, wasteful usage of resources, debugging challenge.
- Solution: Real-Time Memory Allocation Tracker has interactive visualization and monitoring for memory optimization.

Conclusion & Future Scope

- The tool is successful in tracing memory in real-time.
- Future Improvements:
- Implement with real system memory API.
- Improve UI with analytics.
- Offer optimization recommendations.

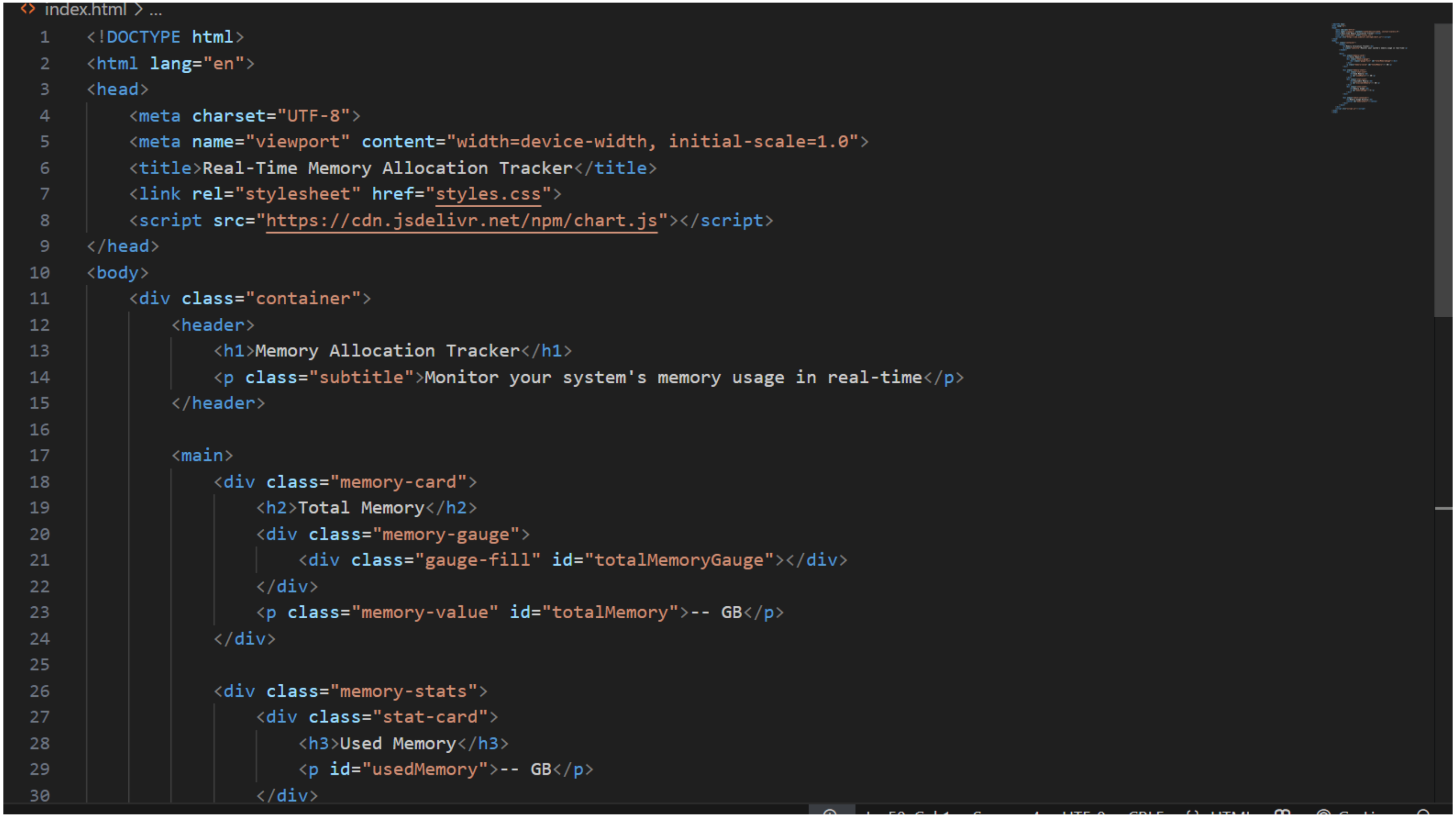
References

- - [GitHub Repository](https://github.com/sauravsuman18/Real-Time-Memory-Allocation-Tracker)

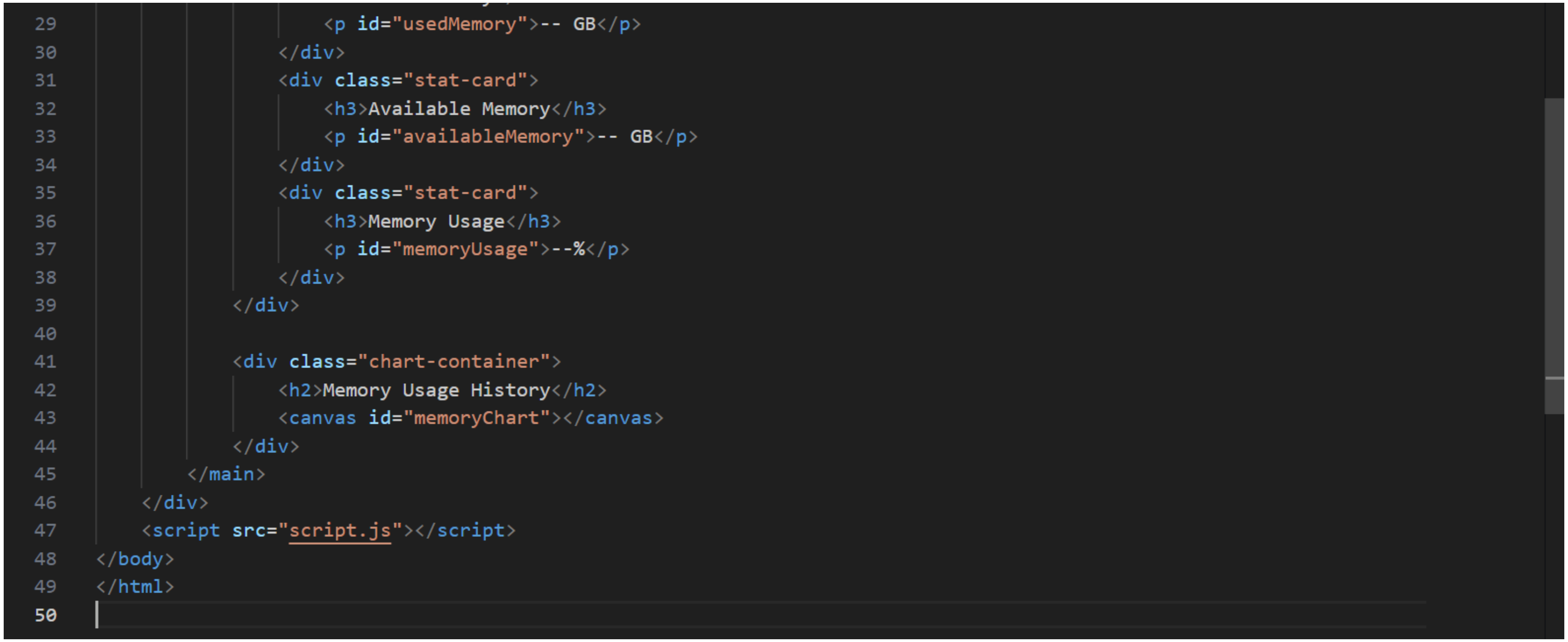
Problem Statement

- Efficient management of memory allocation is an important area of software development, particularly for applications that deal with dynamic data structures and high-performance computing. Inadequate real-time visibility into memory usage can cause performance bottlenecks, wasteful resource usage, and longer debugging time. The Real-Time Memory Allocation Tracker resolves this problem by offering an interactive and visualized view of memory allocation, enabling developers and administrators to monitor and optimize memory usage efficiently.

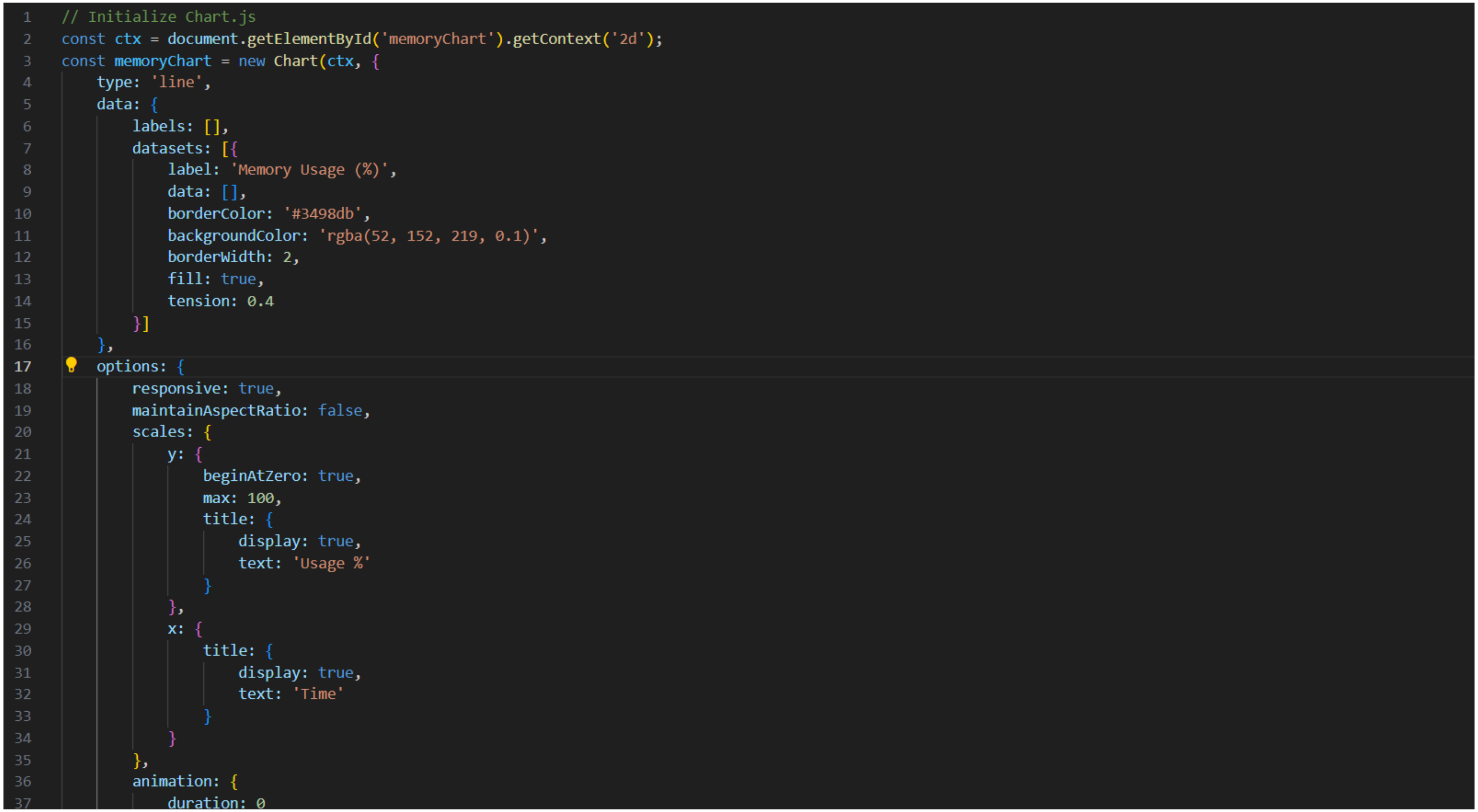
CODE (SOLUTION)



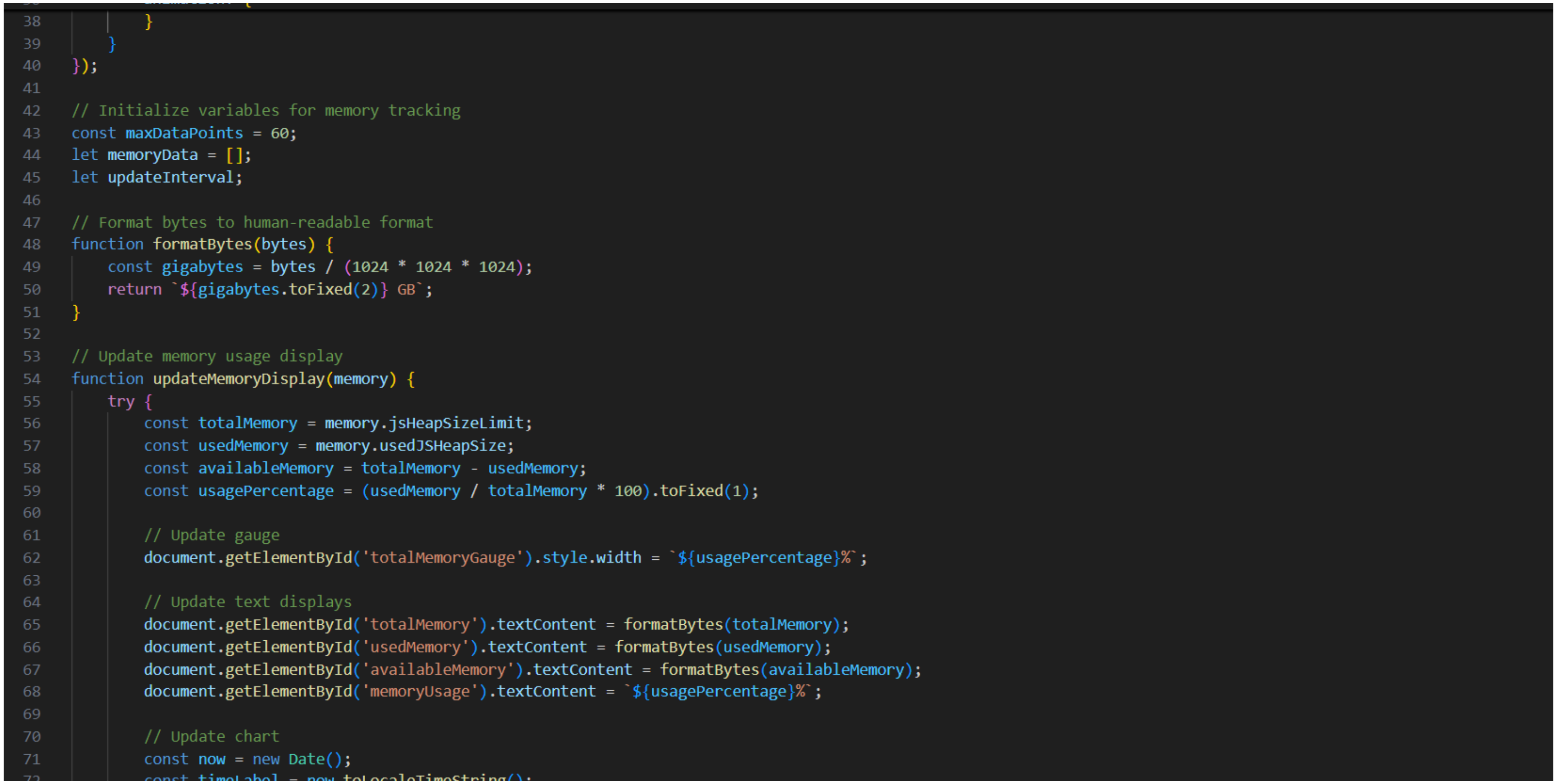
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Real-Time Memory Allocation Tracker</title>
    <link rel="stylesheet" href="styles.css">
    <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
</head>
<body>
    <div class="container">
        <header>
            <h1>Memory Allocation Tracker</h1>
            <p class="subtitle">Monitor your system's memory usage in real-time</p>
        </header>

        <main>
            <div class="memory-card">
                <h2>Total Memory</h2>
                <div class="memory-gauge">
                    <div class="gauge-fill" id="totalMemoryGauge"></div>
                </div>
                <p class="memory-value" id="totalMemory">-- GB</p>
            </div>

            <div class="memory-stats">
                <div class="stat-card">
                    <h3>Used Memory</h3>
                    <p id="usedMemory">-- GB</p>
                </div>
```



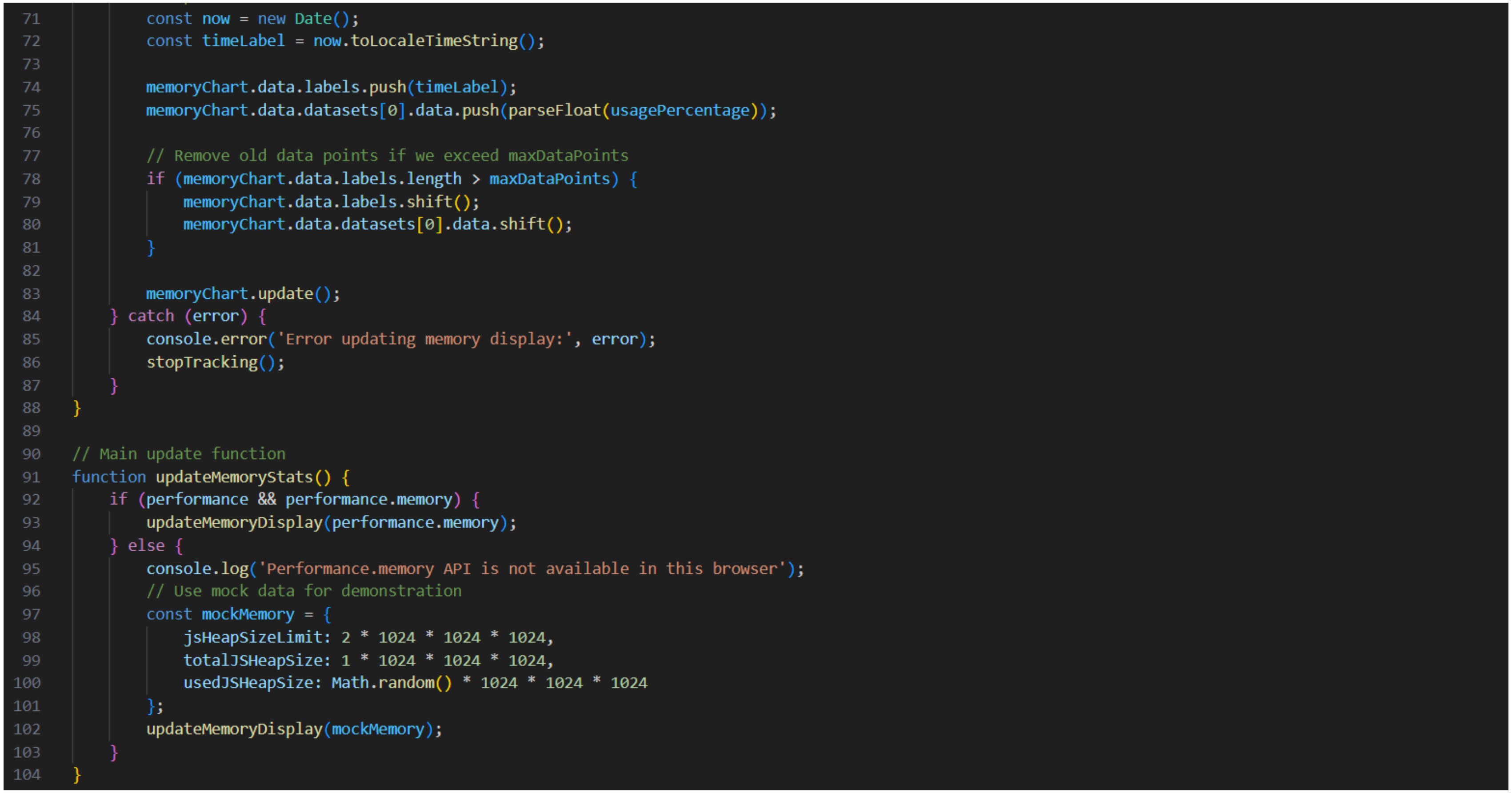
```html
                    <p id="usedMemory">-- GB</p>
                </div>
                <div class="stat-card">
                    <h3>Available Memory</h3>
                    <p id="availableMemory">-- GB</p>
                </div>
                <div class="stat-card">
                    <h3>Memory Usage</h3>
                    <p id="memoryUsage">--%</p>
                </div>
            </div>

            <div class="chart-container">
                <h2>Memory Usage History</h2>
                <canvas id="memoryChart"></canvas>
            </div>
        </main>
    </div>
    <script src="script.js"></script>
</body>
</html>

```



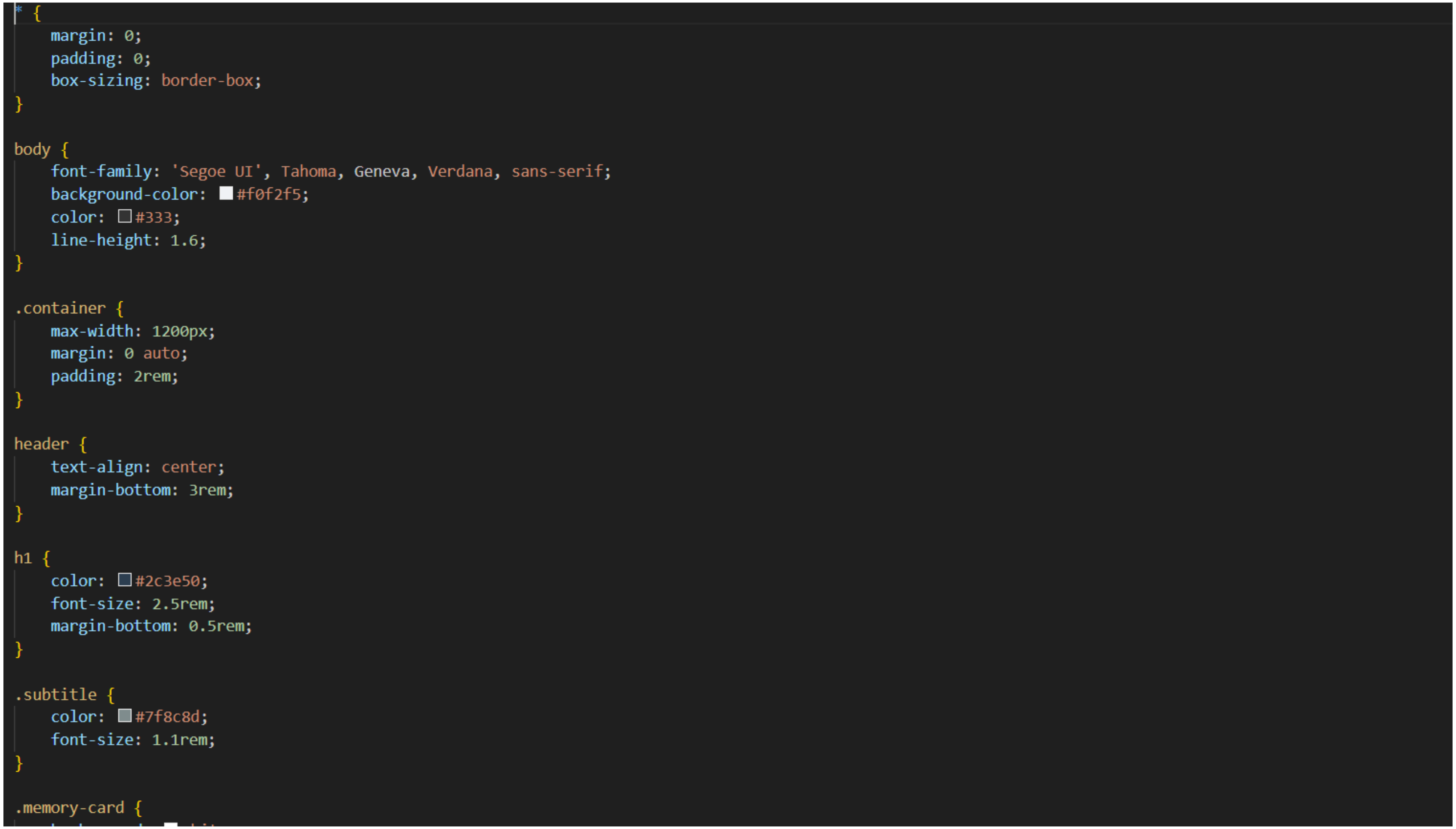
```javascript
// Initialize Chart.js
const ctx = document.getElementById('memoryChart').getContext('2d');
const memoryChart = new Chart(ctx, {
    type: 'line',
    data: {
        labels: [],
        datasets: [{
            label: 'Memory Usage (%)',
            data: [],
            borderColor: '#3498db',
            backgroundColor: 'rgba(52, 152, 219, 0.1)',
            borderWidth: 2,
            fill: true,
            tension: 0.4
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        scales: {
            y: {
                beginAtZero: true,
                max: 100,
                title: {
                    display: true,
                    text: 'Usage %'
                }
            },
            x: {
                title: {
                    display: true,
                    text: 'Time'
                }
            }
        },
        animation: {
            duration: 0
```



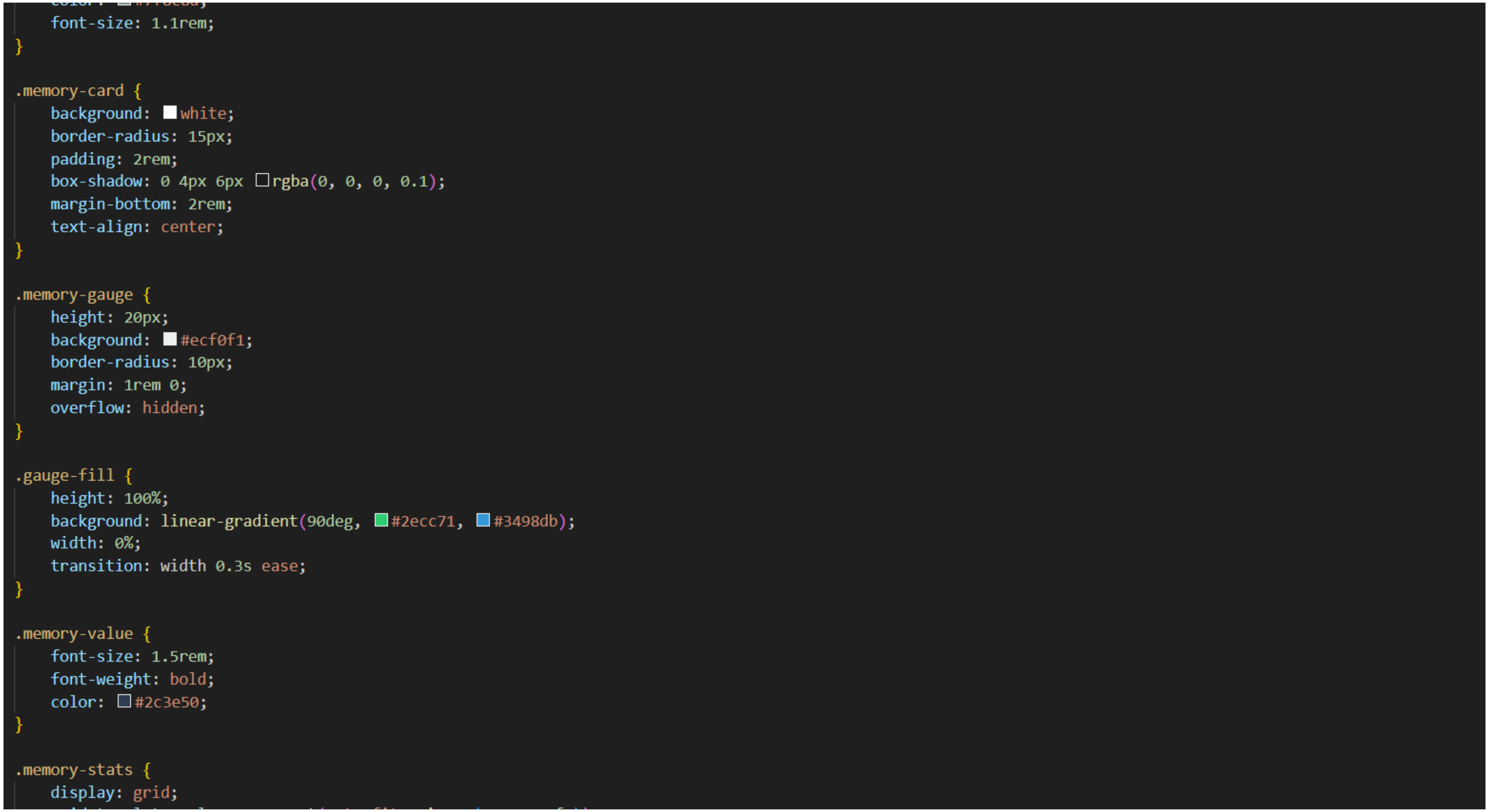
```
        animation: {
        }
    }
});

// Initialize variables for memory tracking
const maxDataPoints = 60;
let memoryData = [];
let updateInterval;

// Format bytes to human-readable format
function formatBytes(bytes) {
    const gigabytes = bytes / (1024 * 1024 * 1024);
    return `${gigabytes.toFixed(2)} GB`;
}

// Update memory usage display
function updateMemoryDisplay(memory) {
    try {
        const totalMemory = memory.jsHeapSizeLimit;
        const usedMemory = memory.usedJSHeapSize;
        const availableMemory = totalMemory - usedMemory;
        const usagePercentage = (usedMemory / totalMemory * 100).toFixed(1);

        // Update gauge
        document.getElementById('totalMemoryGauge').style.width = `${usagePercentage}%`;

        // Update text displays
        document.getElementById('totalMemory').textContent = formatBytes(totalMemory);
        document.getElementById('usedMemory').textContent = formatBytes(usedMemory);
        document.getElementById('availableMemory').textContent = formatBytes(availableMemory);
        document.getElementById('memoryUsage').textContent = `${usagePercentage}%`;

        // Update chart
        const now = new Date();
        const timeLabel = now.toLocaleTimeString();
```



```
        const now = new Date();
        const timeLabel = now.toLocaleTimeString();

        memoryChart.data.labels.push(timeLabel);
        memoryChart.data.datasets[0].data.push(parseFloat(usagePercentage));

        // Remove old data points if we exceed maxDataPoints
        if (memoryChart.data.labels.length > maxDataPoints) {
            memoryChart.data.labels.shift();
            memoryChart.data.datasets[0].data.shift();
        }

        memoryChart.update();
    } catch (error) {
        console.error('Error updating memory display:', error);
        stopTracking();
    }
}

// Main update function
function updateMemoryStats() {
    if (performance && performance.memory) {
        updateMemoryDisplay(performance.memory);
    } else {
        console.log('Performance.memory API is not available in this browser');
        // Use mock data for demonstration
        const mockMemory = {
            jsHeapSizeLimit: 2 * 1024 * 1024 * 1024,
            totalJSHeapSize: 1 * 1024 * 1024 * 1024,
            usedJSHeapSize: Math.random() * 1024 * 1024 * 1024
        };
        updateMemoryDisplay(mockMemory);
    }
}
```



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

body {
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    background-color: #f0f2f5;
    color: #333;
    line-height: 1.6;
}

.container {
    max-width: 1200px;
    margin: 0 auto;
    padding: 2rem;
}

header {
    text-align: center;
    margin-bottom: 3rem;
}

h1 {
    color: #2c3e50;
    font-size: 2.5rem;
    margin-bottom: 0.5rem;
}

.subtitle {
    color: #7f8c8d;
    font-size: 1.1rem;
}

.memory-card {
```



```css
    color: #7f8c8d;
    font-size: 1.1rem;
}

.memory-card {
    background: white;
    border-radius: 15px;
    padding: 2rem;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);
    margin-bottom: 2rem;
    text-align: center;
}

.memory-gauge {
    height: 20px;
    background: #ecf0f1;
    border-radius: 10px;
    margin: 1rem 0;
    overflow: hidden;
}

.gauge-fill {
    height: 100%;
    background: linear-gradient(90deg, #2ecc71, #3498db);
    width: 0%;
    transition: width 0.3s ease;
}

.memory-value {
    font-size: 1.5rem;
    font-weight: bold;
    color: #2c3e50;
}

.memory-stats {
    display: grid;
```