```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sentiment Analysis Dashboard</title>
  <!-- Tailwind CSS -->
  <script src="https://cdn.tailwindcss.com"></script>
  <!-- Chart.js for visualizations -->
  <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
  <!-- jsPDF for PDF export -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/jspdf/2.5.1/jspdf.umd.min.js"></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/jspdf-
autotable/3.5.23/jspdf.plugin.autotable.min.js"></script>
  <!-- Gauge.js for speedometers -->
  <script src="https://bernii.github.io/gauge.js/dist/gauge.min.js"></script>
  <!-- PDF.js for PDF text extraction -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/pdf.js/2.11.338/pdf.min.js"></script>
  <!-- Mammoth.js for DOCX text extraction -->
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/mammoth/1.6.0/mammoth.browser.min.js"
></script>
  <!-- Lucide Icons -->
  <script src="https://unpkg.com/lucide@latest"></script>
  <script>
    // Custom Tailwind theme configuration
    tailwind.config = {
```

```
      theme: {
        extend: {
          colors: {
            'custom-dark-bg': '#0D1B1E',
            'custom-panel-bg': '#122A2E',
            'custom-primary': '#33D1C4',
            'custom-light-text': '#E0F2F1',
            'custom-muted-text': '#AAB8C2',
          }
        }
      }
    }
</script>
<style>
    /* Custom scrollbar for a cleaner look */
    .custom-scrollbar::-webkit-scrollbar {
        width: 8px;
    }
    .custom-scrollbar::-webkit-scrollbar-track {
        background: #122A2E;
    }
    .custom-scrollbar::-webkit-scrollbar-thumb {
        background: #33D1C4;
        border-radius: 4px;
    }
    .custom-scrollbar::-webkit-scrollbar-thumb:hover {
        background: #29a89d;
    }
```

```
.progress-bar-fill {
  transition: width 0.5s ease-in-out; /* Slower, more noticeable transition */
}
@keyframes fadeIn {
  from { opacity: 0; transform: translateY(20px); }
  to { opacity: 1; transform: translateY(0); }
}
.fade-in {
  animation: fadeIn 1s ease-out forwards;
  opacity: 0; /* Start hidden for animation */
}
/* Guided Tour Styles */
#tour-overlay {
  position: fixed;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
  background-color: rgba(0, 0, 0, 0.7);
  z-index: 999;
}
.tour-highlight {
  position: relative;
  z-index: 1000;
  box-shadow: 0 0 0 9999px rgba(0, 0, 0, 0.7);
  border-radius: 8px;
}
#tour-tip {
```

```
        position: fixed; /* Use fixed positioning for tips */
        background-color: #122A2E;
        color: #E0F2F1;
        padding: 1.5rem;
        border-radius: 8px;
        border: 1px solid #33D1C4;
        box-shadow: 0 4px 12px rgba(0,0,0,0.15);
        z-index: 1001;
        width: 320px;
    }
  </style>
</head>
<body class="bg-custom-dark-bg font-sans text-custom-light-text">

  <!-- Landing Page -->
  <div id="landing-page" class="min-h-screen bg-gradient-to-br from-custom-dark-bg to-gray-900 text-white flex flex-col justify-center items-center p-8">
    <div class="text-center max-w-3xl">
      <h1 class="text-5xl md:text-7xl font-bold mb-4 fade-in text-custom-primary" style="animation-delay: 0.2s;">Discover the Emotion in Your Text</h1>
      <p class="text-xl md:text-2xl mb-8 fade-in text-custom-light-text" style="animation-delay: 0.4s;">Instantly reveal the emotional tone of your reviews, messages, or documents.</p>
      <button id="start-analyzing-btn" class="bg-custom-primary text-custom-dark-bg font-bold py-3 px-8 rounded-full text-lg hover:bg-opacity-80 transition-transform transform hover:scale-105 fade-in" style="animation-delay: 0.6s;">
        Analyze Now
      </button>
    </div>
  </div>
```

```
<!-- Dashboard -->
<div id="dashboard" class="hidden max-w-7xl mx-auto p-4 sm:p-6 lg:p-8">
  <header class="mb-8 flex justify-between items-start">
    <div>
      <h1 class="text-4xl font-bold text-white">Sentiment Analysis Dashboard</h1>
      <p class="text-lg text-custom-muted-text mt-1">Understand emotional tone in your text data instantly.</p>
    </div>
    <button id="clear-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-custom-panel-bg hover:text-red-500 transition-colors flex-shrink-0">
      <i data-lucide="refresh-cw" class="h-4 w-4 mr-2"></i> Refresh & Clear
    </button>
  </header>

  <!-- Error Display -->
  <div id="error-container" class="hidden bg-red-900 border-l-4 border-red-500 text-red-200 p-4 rounded-lg mb-6 justify-between items-center shadow-md">
    <div class="flex items-center">
      <i data-lucide="alert-circle" class="h-6 w-6 mr-3"></i>
      <span id="error-message"></span>
    </div>
    <button id="close-error-btn" class="text-red-300 hover:text-red-100">
      <i data-lucide="x" class="h-5 w-5"></i>
    </button>
  </div>

  <!-- View Toggles -->
```

```
<div class="flex items-center space-x-2 mb-6 border-b border-gray-700">
  <button id="view-dashboard-btn" class="px-4 py-2 text-sm font-medium border-b-2 border-custom-primary text-custom-light-text">
    Dashboard
  </button>
  <button id="view-comparison-btn" class="px-4 py-2 text-sm font-medium text-custom-muted-text hover:text-white">
    Comparison (<span id="comparison-count">0</span>)
  </button>
  <button id="view-gemini-features-btn" class="px-4 py-2 text-sm font-medium text-custom-muted-text hover:text-white">
    2 Point creativity features
  </button>
</div>

<!-- Main Content Area -->
<main id="main-content">
  <!-- Dashboard View -->
  <div id="dashboard-view">
    <!-- Key Metrics Widgets -->
    <div class="grid grid-cols-1 md:grid-cols-3 gap-6 mb-6">
      <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700 flex items-center space-x-4 fade-in" style="animation-delay: 0.2s;">
        <i data-lucide="smile" class="h-12 w-12 text-green-500"></i>
        <div>
          <p class="text-custom-muted-text">Total Positive</p>
          <h2 id="positive-metric" class="text-4xl font-bold text-white">0</h2>
        </div>
      </div>
```

```
        <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700 flex items-center space-x-4 fade-in" style="animation-delay: 0.4s;">
          <i data-lucide="frown" class="h-12 w-12 text-red-500"></i>
          <div>
            <p class="text-custom-muted-text">Total Negative</p>
            <h2 id="negative-metric" class="text-4xl font-bold text-white">0</h2>
          </div>
        </div>
        <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700 flex items-center space-x-4 fade-in" style="animation-delay: 0.6s;">
          <i data-lucide="meh" class="h-12 w-12 text-gray-500"></i>
          <div>
            <p class="text-custom-muted-text">Total Neutral</p>
            <h2 id="neutral-metric" class="text-4xl font-bold text-white">0</h2>
          </div>
        </div>
      </div>

      <div class="grid grid-cols-1 lg:grid-cols-3 gap-6">
        <!-- Left Column -->
        <div class="lg:col-span-2 space-y-6">
          <!-- Input Section -->
          <div id="tour-step-1" class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700 fade-in" style="animation-delay: 0.8s;">
            <h2 class="text-xl font-semibold text-white mb-4">Analyze New Text</h2>
            <textarea id="text-input" placeholder="Enter text here for sentiment analysis..." class="w-full h-24 p-3 bg-custom-dark-bg border border-gray-600 rounded-lg text-custom-light-text focus:ring-2 focus:ring-custom-primary focus:border-custom-primary transition"></textarea>
            <div class="mt-4 flex items-center justify-between">
```

```
                <div class="flex items-center space-x-3">
                    <button id="upload-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-gray-700 flex-shrink-0">
                        <i data-lucide="upload" class="h-4 w-4 mr-2"></i> Upload File(s)
                    </button>
                    <span id="file-name-display" class="text-sm text-custom-muted-text truncate" title="No file selected">No file selected</span>
                    <input type="file" id="file-input" class="hidden" accept=".txt,.csv,.md,.json,.pdf,.docx" multiple>
                </div>
                <div class="flex items-center space-x-2">
                    <button id="cancel-btn" class="hidden px-5 py-2.5 bg-red-600 text-white font-semibold rounded-lg hover:bg-red-700 flex items-center transition-colors">
                        Cancel
                    </button>
                    <button id="analyze-btn" class="px-5 py-2.5 bg-custom-primary text-custom-dark-bg font-semibold rounded-lg hover:bg-opacity-80 disabled:bg-opacity-50 disabled:cursor-not-allowed flex items-center transition-colors">
                        Analyze
                    </button>
                </div>
            </div>
            <!-- Progress Bar -->
            <div id="progress-container" class="hidden mt-4">
                <div class="flex justify-between mb-1">
                    <span class="text-base font-medium text-custom-primary">Analysis Progress</span>
                    <span id="progress-text" class="text-sm font-medium text-custom-primary">0%</span>
                </div>
```

```
                    <div class="w-full bg-gray-700 rounded-full h-2.5">
                        <div id="progress-bar" class="bg-custom-primary h-2.5 rounded-full progress-bar-fill" style="width: 0%"></div>
                    </div>
                </div>
            </div>
            <!-- Uploaded Files Display -->
            <div id="uploaded-files-container" class="hidden bg-custom-panel-bg p-4 rounded-2xl shadow-lg border border-gray-700">
                <h3 class="text-md font-semibold text-white mb-2">Uploaded Files</h3>
                <ul id="uploaded-files-list" class="text-custom-muted-text space-y-1"></ul>
            </div>
        </div>

        <!-- Right Column: Sentiment Distribution -->
        <div id="tour-step-2" class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700 fade-in" style="animation-delay: 1s;">
            <h2 class="text-xl font-semibold text-white mb-4">Sentiment Distribution</h2>
            <div id="visuals-container" class="space-y-6">
                <!-- Gauges -->
                <div class="grid grid-cols-3 gap-4 text-center">
                    <div>
                        <canvas id="positive-gauge" class="w-full h-auto"></canvas>
                        <p class="text-sm font-semibold text-green-500 mt-1">Positive</p>
                    </div>
                    <div>
                        <canvas id="negative-gauge" class="w-full h-auto"></canvas>
```

```
                    <p class="text-sm font-semibold text-red-500 mt-1">Negative</p>
                  </div>
                  <div>
                    <canvas id="neutral-gauge" class="w-full h-auto"></canvas>
                    <p class="text-sm font-semibold text-gray-500 mt-1">Neutral</p>
                  </div>
                </div>
                <!-- Pie Chart -->
                <div class="h-[200px]"><canvas id="sentiment-pie-chart"></canvas></div>
                 <!-- Bar Chart -->
                <div class="h-[200px]"><canvas id="sentiment-bar-chart"></canvas></div>
              </div>
              <div id="visuals-placeholder" class="h-full flex flex-col items-center justify-center text-center text-custom-muted-text">
                <i data-lucide="bar-chart-2" class="h-16 w-16 mb-4 mx-auto"></i>
                <p>No data to display. Analyze some text to see the distribution.</p>
              </div>
            </div>
          </div>

          <!-- Results Section -->
          <div id="tour-step-3" class="mt-8 bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700 fade-in" style="animation-delay: 1.2s;">
            <div class="flex justify-between items-center mb-4">
              <h2 class="text-xl font-semibold text-white">Analysis Results</h2>
               <div class="relative">
```

```
                <button id="export-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-gray-700">
                    <i data-lucide="share-2" class="h-4 w-4 mr-2"></i> Export <i data-lucide="chevron-down" class="h-4 w-4 ml-1"></i>
                </button>
                <div id="export-menu" class="absolute hidden right-0 mt-2 w-40 bg-custom-panel-bg rounded-lg shadow-xl z-10 border border-gray-600">
                    <a href="#" id="export-csv" class="block px-4 py-2 text-sm text-custom-muted-text hover:bg-gray-700">Export as CSV</a>
                    <a href="#" id="export-json" class="block px-4 py-2 text-sm text-custom-muted-text hover:bg-gray-700">Export as JSON</a>
                    <a href="#" id="export-pdf" class="block px-4 py-2 text-sm text-custom-muted-text hover:bg-gray-700">Export as PDF Report</a>
                </div>
            </div>
        </div>
        <div id="results-container" class="max-h-[50vh] overflow-y-auto pr-2 space-y-4 custom-scrollbar">
            <div id="results-placeholder" class="text-center py-12 text-custom-muted-text">
                <i data-lucide="file-text" class="mx-auto h-12 w-12 mb-4"></i>
                <p>Your analyzed texts will appear here.</p>
            </div>
        </div>
    </div>
</div>

<!-- Comparison View -->
<div id="comparison-view" class="hidden bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700">
    <h2 class="text-xl font-semibold text-white mb-4">Comparative Analysis</h2>
```

```
        <div id="comparison-container" class="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-6">
            <div id="comparison-placeholder" class="text-center py-12 text-custom-muted-text md:col-span-2 lg:col-span-3">
                <i data-lucide="columns" class="mx-auto h-12 w-12 mb-4"></i>
                <p>Select 2 or more texts from the results list to compare them.</p>
            </div>
        </div>
    </div>

    <!-- Gemini Features View -->
    <div id="gemini-features-view" class="hidden space-y-6">
        <div class="grid grid-cols-1 md:grid-cols-2 gap-6">
            <!-- Summary & Themes -->
            <div class="space-y-6">
                <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700">
                    <div class="flex justify-between items-center mb-4">
                        <h2 class="text-xl font-semibold text-white">Automated Summary</h2>
                        <button id="generate-summary-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-gray-700">
                            <i data-lucide="zap" class="h-4 w-4 mr-2"></i> Generate
                        </button>
                    </div>
                    <div id="summary-display" class="text-custom-muted-text prose">
                        <p>Click "Generate" to create an executive summary of your analyzed data.</p>
                    </div>
                </div>
```

```
            <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700">
                <div class="flex justify-between items-center mb-4">
                    <h2 class="text-xl font-semibold text-white">Thematic Analysis</h2>
                    <button id="analyze-themes-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-gray-700">
                        <i data-lucide="zap" class="h-4 w-4 mr-2"></i> Analyze
                    </button>
                </div>
                <div id="themes-display" class="text-custom-muted-text prose">
                    <p>Click "Analyze" to identify recurring themes in the text.</p>
                </div>
            </div>
        </div>
        <!-- Insights & Suggestions -->
        <div class="space-y-6">
            <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700">
                <h2 class="text-xl font-semibold text-white mb-4">Deeper Insights</h2>
                <textarea id="insights-question" placeholder="Ask a question about your data, e.g., 'What are the main complaints?'" class="w-full h-20 p-3 bg-custom-dark-bg border border-gray-600 rounded-lg text-custom-light-text focus:ring-2 focus:ring-custom-primary focus:border-custom-primary transition"></textarea>
                <div class="text-right mt-2">
                    <button id="get-insights-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-gray-700 ml-auto">
                        <i data-lucide="zap" class="h-4 w-4 mr-2"></i> Get Insights
                    </button>
                </div>
```

```
                <div id="insights-display" class="text-custom-muted-text mt-4 prose">
                  <p>Ask a question to get specific insights from your data.</p>
                </div>
              </div>
              <div class="bg-custom-panel-bg p-6 rounded-2xl shadow-lg border border-gray-700">
                <div class="flex justify-between items-center mb-4">
                  <h2 class="text-xl font-semibold text-white">Actionable Suggestions</h2>
                  <button id="suggest-actions-btn" class="flex items-center px-4 py-2 border border-gray-600 rounded-lg text-sm font-medium text-custom-muted-text hover:bg-gray-700">
                    <i data-lucide="zap" class="h-4 w-4 mr-2"></i> Suggest
                  </button>
                </div>
                <div id="actions-display" class="text-custom-muted-text prose">
                  <p>Click "Suggest" to generate actionable steps based on the analysis.</p>
                </div>
              </div>
            </div>
          </div>
        </main>
      </div>

      <!-- Tour Modal -->
      <div id="tour-modal" class="hidden fixed inset-0 bg-gray-900 bg-opacity-50 flex items-center justify-center z-50">
        <div class="bg-white rounded-lg shadow-xl p-8 max-w-sm text-center">
```

```
        <h2 class="text-2xl font-bold mb-4 text-gray-800">Welcome!</h2>
        <p class="text-gray-600 mb-6">Would you like a quick tour of the features?</p>
        <div class="flex justify-center space-x-4">
            <button id="tour-yes-btn" class="bg-custom-primary text-white font-semibold py-2 px-6 rounded-lg hover:bg-custom-dark">Yes, please!</button>
            <button id="tour-no-btn" class="bg-gray-200 text-gray-800 font-semibold py-2 px-6 rounded-lg hover:bg-gray-300">No, thanks</button>
        </div>
    </div>
</div>



<script>
    // Required for PDF.js to work
    pdfjsLib.GlobalWorkerOptions.workerSrc = `https://cdnjs.cloudflare.com/ajax/libs/pdf.js/2.11.338/pdf.worker.min.js`;

    document.addEventListener('DOMContentLoaded', () => {
        // --- STATE MANAGEMENT ---
        const initialState = {
            texts: [],
            uploadedFiles: [],
            isLoading: false,
            isGenerating: { summary: false, themes: false, insights: false, actions: false },
            error: null,
            view: 'dashboard',
            comparisonSelection: [],
            cancellationRequested: false,
            progress: 0,
```

```
    geminiResults: { summary: '', themes: '', insights: '', actions: '' },
};
let state = { ...initialState };

// --- DOM ELEMENTS ---
const landingPage = document.getElementById('landing-page');
const dashboard = document.getElementById('dashboard');
const startAnalyzingBtn = document.getElementById('start-analyzing-btn');
const textInput = document.getElementById('text-input');
const analyzeBtn = document.getElementById('analyze-btn');
const cancelBtn = document.getElementById('cancel-btn');
const clearBtn = document.getElementById('clear-btn');
const uploadBtn = document.getElementById('upload-btn');
const fileInput = document.getElementById('file-input');
const fileNameDisplay = document.getElementById('file-name-display');
const uploadedFilesContainer = document.getElementById('uploaded-files-container');
const uploadedFilesList = document.getElementById('uploaded-files-list');
const resultsContainer = document.getElementById('results-container');
const resultsPlaceholder = document.getElementById('results-placeholder');
const visualsContainer = document.getElementById('visuals-container');
const visualsPlaceholder = document.getElementById('visuals-placeholder');
const errorContainer = document.getElementById('error-container');
const errorMessage = document.getElementById('error-message');
const closeErrorBtn = document.getElementById('close-error-btn');
const viewDashboardBtn = document.getElementById('view-dashboard-btn');
const viewComparisonBtn = document.getElementById('view-comparison-btn');
const viewGeminiFeaturesBtn = document.getElementById('view-gemini-features-btn');
```

```
        const dashboardView = document.getElementById('dashboard-view');
        const comparisonView = document.getElementById('comparison-view');
        const geminiFeaturesView = document.getElementById('gemini-features-view');
        const comparisonContainer = document.getElementById('comparison-container');
        const comparisonPlaceholder = document.getElementById('comparison-placeholder');
        const comparisonCount = document.getElementById('comparison-count');
        const progressContainer = document.getElementById('progress-container');
        const progressBar = document.getElementById('progress-bar');
        const progressText = document.getElementById('progress-text');
        const tourModal = document.getElementById('tour-modal');
        const tourYesBtn = document.getElementById('tour-yes-btn');
        const tourNoBtn = document.getElementById('tour-no-btn');
        const exportBtn = document.getElementById('export-btn');
        const exportMenu = document.getElementById('export-menu');
        const positiveMetric = document.getElementById('positive-metric');
        const negativeMetric = document.getElementById('negative-metric');
        const neutralMetric = document.getElementById('neutral-metric');
        // Gemini Feature Elements
        const generateSummaryBtn = document.getElementById('generate-summary-btn');
        const summaryDisplay = document.getElementById('summary-display');
        const analyzeThemesBtn = document.getElementById('analyze-themes-btn');
        const themesDisplay = document.getElementById('themes-display');
        const insightsQuestion = document.getElementById('insights-question');
        const getInsightsBtn = document.getElementById('get-insights-btn');
        const insightsDisplay = document.getElementById('insights-display');
        const suggestActionsBtn = document.getElementById('suggest-actions-btn');
```

```
const actionsDisplay = document.getElementById('actions-display');


// --- ICONS ---
if (typeof lucide !== 'undefined') {
  lucide.createIcons();
}


// --- CHART & GAUGE INSTANCES ---
let sentimentPieChart = null;
let sentimentBarChart = null;
let gauges = { Positive: null, Negative: null, Neutral: null };


// --- SENTIMENT CONFIG ---
const SENTIMENT_CONFIG = {
  Positive: { color: '#22c55e', emoji: '😊' },
  Negative: { color: '#ef4444', emoji: '😟' },
  Neutral: { color: '#64748b', emoji: '😐' }
};


// --- API FUNCTIONS ---
const callGeminiApi = async (text) => {
  await new Promise(resolve => setTimeout(resolve, 1000 + Math.random() *
500));
  const prompt = `Perform a sentiment analysis on the following text. Your
primary goal is to determine if the emotional tone is Positive, Negative, or Neutral. Be
decisive: if there is any clear emotional language, avoid defaulting to Neutral. Your
response MUST be a JSON object with the following structure: { "sentiment": "...",
"confidence": ..., "keywords": ["...", "..."], "explanation": "..." }. Here is the text to analyze:
"${text}"`;
  const chatHistory = [{ role: "user", parts: [{ text: prompt }] }];
```

```
        const payload = {
          contents: chatHistory,
          generationConfig: {
            responseMimeType: "application/json",
            responseSchema: {
              type: "OBJECT",
              properties: {
                "sentiment": { "type": "STRING" },
                "confidence": { "type": "NUMBER" },
                "keywords": { "type": "ARRAY", "items": { "type": "STRING" } },
                "explanation": { "type": "STRING" }
              },
              required: ["sentiment", "confidence", "keywords", "explanation"]
            }
          }
        };
        const apiKey = "AIzaSyDT4qHLHTMpcatOyTeUs0ywnnOBybs603o";
        const apiUrl =
`https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash-preview-05-20:generateContent?key=${apiKey}`;

        let retries = 3;
        let delay = 1000;
        for (let i = 0; i < retries; i++) {
          try {
            const response = await fetch(apiUrl, {
              method: 'POST',
              headers: { 'Content-Type': 'application/json' },
              body: JSON.stringify(payload)
```

```
            });

            if (response.ok) {
                const result = await response.json();
                if (result.candidates && result.candidates[0].content &&
result.candidates[0].content.parts[0]) {
                    return JSON.parse(result.candidates[0].content.parts[0].text);
                }
            } else {
                if (response.status === 400 || response.status === 403) {
                    const errorResult = await response.json();
                    const errorMessage = errorResult?.error?.message || `API Error:
${response.statusText}. Check API Key and permissions.`;
                    throw new Error(errorMessage);
                }
                console.error(`API request failed with status ${response.status},
retrying...`);
            }
        } catch (error) {
            console.error("API call failed:", error);
            if (i === retries - 1) {
                throw error;
            }
        }
        await new Promise(resolve => setTimeout(resolve, delay));
        delay *= 2;
    }
    throw new Error("Failed to analyze sentiment after multiple retries.");
};
```

```
const generateGeminiText = async (prompt) => {
    const chatHistory = [{ role: "user", parts: [{ text: prompt }] }];
    const payload = { contents: chatHistory };
    const apiKey = "AIzaSyDT4qHLHTMpcatOyTeUs0ywnnOBybs603o";
    const apiUrl =
`https://generativelanguage.googleapis.com/v1beta/models/gemini-2.5-flash-preview-05-20:generateContent?key=${apiKey}`;

    try {
      const response = await fetch(apiUrl, {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify(payload)
      });
      if (!response.ok) {
        throw new Error(`API Error: ${response.statusText}`);
      }
      const result = await response.json();
      if (result.candidates && result.candidates[0].content &&
result.candidates[0].content.parts[0]) {
        return result.candidates[0].content.parts[0].text;
      }
    } catch (error) {
      console.error("Gemini text generation failed:", error);
      throw error;
    }
    return null;
  };
```

```
// --- HELPER FUNCTIONS ---
function chunkText(text, maxLength = 5000) {
  const chunks = [];
  if (!text) return chunks;
  const sentences = text.match(/[^.!?]+[.!?]*/g) || [text];
  let currentChunk = "";
  for (const sentence of sentences) {
    if (currentChunk.length + sentence.length > maxLength) {
      chunks.push(currentChunk.trim());
      currentChunk = "";
    }
    currentChunk += sentence + " ";
  }
  if (currentChunk.trim().length > 0) {
    chunks.push(currentChunk.trim());
  }
  return chunks;
}

function animateCounter(element, end, duration = 1000) {
  let start = 0;
  const finalEnd = parseInt(end, 10);
  if (element.textContent) {
    start = parseInt(element.textContent, 10);
  }
  let startTimestamp = null;
  const step = (timestamp) => {
```

```
    if (!startTimestamp) startTimestamp = timestamp;
    const progress = Math.min((timestamp - startTimestamp) / duration, 1);
    element.textContent = Math.floor(progress * (finalEnd - start) + start);
    if (progress < 1) {
      window.requestAnimationFrame(step);
    }
  };
  window.requestAnimationFrame(step);
}

// --- RENDER FUNCTIONS ---
const render = () => {
  try {
    // Render loading state
    analyzeBtn.disabled = state.isLoading || !textInput.value.trim();
    textInput.disabled = state.isLoading;
    if (state.isLoading) {
      analyzeBtn.classList.add('hidden');
      cancelBtn.classList.remove('hidden');
      progressContainer.classList.remove('hidden');
    } else {
      analyzeBtn.classList.remove('hidden');
      cancelBtn.classList.add('hidden');
      progressContainer.classList.add('hidden');
    }

    // Render error
    if (state.error) {
```

```
            errorMessage.textContent = state.error;
            errorContainer.style.display = 'flex';
        } else {
            errorContainer.style.display = 'none';
        }

        // Render views
        const views = {
            dashboard: dashboardView,
            comparison: comparisonView,
            gemini: geminiFeaturesView
        };
        const buttons = {
            dashboard: viewDashboardBtn,
            comparison: viewComparisonBtn,
            gemini: viewGeminiFeaturesBtn
        };

        Object.keys(views).forEach(key => {
            views[key].style.display = state.view === key ? 'block' : 'none';
            buttons[key].className = state.view === key
                ? 'px-4 py-2 text-sm font-medium border-b-2 border-custom-primary text-custom-light-text'
                : 'px-4 py-2 text-sm font-medium text-custom-muted-text hover:text-white';
        });

        if(state.view === 'comparison') renderComparison();
        if(state.view === 'gemini') renderGeminiFeatures();
```

```
        renderResults();
        renderVisuals();
        renderProgress();
        renderUploadedFiles();
    } catch (e) {
        console.error("Render error:", e);
        state.error = "A UI error occurred. Please refresh the page.";
        errorMessage.textContent = state.error;
        errorContainer.style.display = 'flex';
    } finally {
        try {
            if (typeof lucide !== 'undefined') {
                lucide.createIcons();
            }
        } catch (e) {
            console.error("Icon rendering failed:", e);
        }
    }
};

const renderProgress = () => {
    const percentage = Math.round(state.progress);
    progressBar.style.width = `${percentage}%`;
    progressText.textContent = `${percentage}%`;
};

const renderUploadedFiles = () => {
```

```
        if(state.uploadedFiles.length === 0) {
            uploadedFilesContainer.classList.add('hidden');
            return;
        }
        uploadedFilesContainer.classList.remove('hidden');
        uploadedFilesList.innerHTML = '';
        state.uploadedFiles.forEach(file => {
            const li = document.createElement('li');
            li.className = 'flex items-center justify-between';
            li.innerHTML = `
                <span>${file.name}</span>
                <button data-file-id="${file.id}" class="delete-file-btn text-red-500 hover:text-red-700">
                    <i data-lucide="trash-2" class="h-4 w-4"></i>
                </button>
            `;
            uploadedFilesList.appendChild(li);
        });
    };

    const renderResults = () => {
        if (state.texts.length === 0) {
            resultsPlaceholder.style.display = 'block';
            resultsContainer.innerHTML = '';
            resultsContainer.appendChild(resultsPlaceholder);
            return;
        }
        resultsPlaceholder.style.display = 'none';
```

```
        resultsContainer.innerHTML = state.texts.map(text => {
            const config = SENTIMENT_CONFIG[text.sentiment] || { color: '#cccccc', emoji: '?' };
            const keywordsHTML = text.keywords.map(kw => `<span class="px-2 py-1 bg-custom-dark text-custom-light-text text-xs font-medium rounded-full">${kw}</span>`).join('');
            const isChecked = state.comparisonSelection.includes(text.id);

            return `
                <div class="p-4 border border-gray-700 rounded-lg bg-custom-panel-bg">
                    <div class="flex justify-between items-start">
                        <p class="text-custom-muted-text flex-1 pr-4">"${text.content}"</p>
                        <div class="flex items-center space-x-4">
                            <div class="flex items-center">
                                <div class="relative flex group">
                                    <input type="checkbox" data-id="${text.id}" ${isChecked ? 'checked' : ''} class="comparison-checkbox h-5 w-5 rounded border-gray-600 text-custom-primary focus:ring-custom-primary cursor-pointer bg-custom-panel-bg"/>
                                    <span class="absolute bottom-full mb-2 w-max px-2 py-1 text-xs text-white bg-gray-800 rounded-md opacity-0 group-hover:opacity-100 transition-opacity duration-300 pointer-events-none">Select for comparison</span>
                                </div>
                            </div>
                            <div class="text-right font-semibold text-lg" style="color: ${config.color};">
                                ${config.emoji} ${text.sentiment}
                            </div>
                            <div class="text-right text-custom-muted-text w-24">
                                <div class="font-semibold text-white">Confidence</div>
                                <div>${(text.confidence * 100).toFixed(1)}%</div>
```

```
                    </div>
                  </div>
                </div>
                <div class="mt-3 pt-3 border-t border-gray-700">
                  <p class="text-sm text-custom-muted-text"><strong class="font-semibold text-white">Explanation:</strong> ${text.explanation}</p>
                  <div class="mt-2">
                    <strong class="text-sm font-semibold text-white">Keywords:</strong>
                    <div class="flex flex-wrap gap-2 mt-1">${keywordsHTML}</div>
                  </div>
                </div>
              </div>
            `;
        }).join('');
    };

    const renderComparison = () => {
        const comparisonData = state.texts.filter(t => state.comparisonSelection.includes(t.id));
        comparisonCount.textContent = state.comparisonSelection.length;

        if (comparisonData.length < 2) {
            comparisonContainer.innerHTML = '';
            comparisonContainer.appendChild(comparisonPlaceholder);
            comparisonPlaceholder.style.display = 'block';
        } else {
            comparisonPlaceholder.style.display = 'none';
            comparisonContainer.innerHTML = comparisonData.map(text => {
```

```
            const config = SENTIMENT_CONFIG[text.sentiment] || { color: '#cccccc', emoji: '?' };
            const keywordsHTML = text.keywords.map(kw => `<span class="px-2 py-1 bg-custom-dark text-custom-light-text text-xs font-medium rounded-full">${kw}</span>`).join('');
            return `
              <div class="p-4 border border-gray-700 rounded-lg bg-custom-panel-bg">
                <div class="font-semibold text-lg mb-2" style="color: ${config.color};">
                  ${config.emoji} ${text.sentiment} (${(text.confidence * 100).toFixed(1)}%)
                </div>
                <p class="text-custom-muted-text text-sm italic mb-3">"${text.content}"</p>
                <div class="mt-3 pt-3 border-t border-gray-700">
                  <p class="text-sm text-custom-muted-text"><strong class="font-semibold text-white">Explanation:</strong> ${text.explanation}</p>
                  <div class="mt-2">
                    <strong class="text-sm font-semibold text-white">Keywords:</strong>
                    <div class="flex flex-wrap gap-2 mt-1">${keywordsHTML}</div>
                  </div>
                </div>
              </div>
            `;
          }).join('');
        }
      };

      const renderVisuals = () => {
        const distribution = { Positive: 0, Negative: 0, Neutral: 0 };
```

```
state.texts.forEach(text => {
  if (distribution[text.sentiment] !== undefined) distribution[text.sentiment]++;
});

// Animate Key Metrics
animateCounter(positiveMetric, distribution.Positive);
animateCounter(negativeMetric, distribution.Negative);
animateCounter(neutralMetric, distribution.Neutral);

if (state.texts.length === 0) {
  visualsPlaceholder.style.display = 'flex';
  visualsContainer.style.display = 'none';
  if (sentimentPieChart) sentimentPieChart.destroy();
  if (sentimentBarChart) sentimentBarChart.destroy();
  sentimentPieChart = null;
  sentimentBarChart = null;
  return;
}

visualsPlaceholder.style.display = 'none';
visualsContainer.style.display = 'block';

const total = state.texts.length;
const chartData = {
  labels: Object.keys(distribution),
  datasets: [{
    data: Object.values(distribution),
```

```
            backgroundColor: Object.keys(distribution).map(key =>
SENTIMENT_CONFIG[key].color),
            hoverOffset: 4
        }]
    };

    if (sentimentPieChart) {
        sentimentPieChart.data = chartData;
        sentimentPieChart.update();
    } else {
        sentimentPieChart = new Chart(document.getElementById('sentiment-pie-
chart'), {
            type: 'pie', data: chartData, options: { responsive: true, maintainAspectRatio:
false, plugins: { legend: { position: 'top', labels: { color: '#E0F2F1' } } } }
        });
    }

    if (sentimentBarChart) {
        sentimentBarChart.data = chartData;
        sentimentBarChart.update();
    } else {
        sentimentBarChart = new Chart(document.getElementById('sentiment-bar-
chart'), {
            type: 'bar', data: chartData, options: { responsive: true,
maintainAspectRatio: false, indexAxis: 'y', plugins: { legend: { display: false } }, scales: { x:
{ ticks: { color: '#E0F2F1' } }, y: { ticks: { color: '#E0F2F1' } } } }
        });
    }
```

```
const gaugeOptions = { angle: 0.15, lineWidth: 0.44, radiusScale: 1, pointer: { length: 0.6, strokeWidth: 0.035, color: '#E0F2F1' }, limitMax: false, limitMin: false, strokeColor: '#AAB8C2', generateGradient: true, highDpiSupport: true };

['Positive', 'Negative', 'Neutral'].forEach(sentiment => {
    const percentage = total > 0 ? (distribution[sentiment] / total) * 100 : 0;
    const canvasId = `${sentiment.toLowerCase()}-gauge`;
    if (!gauges[sentiment]) {
        const target = document.getElementById(canvasId);
        gauges[sentiment] = new Gauge(target).setOptions({ ...gaugeOptions, staticLabels: { font: "12px sans-serif", labels: [0, 50, 100], color: "#E0F2F1" }, staticZones: [{strokeStyle: SENTIMENT_CONFIG[sentiment].color, min: 0, max: 100}] });
    }
    gauges[sentiment].maxValue = 100;
    gauges[sentiment].set(percentage);
});
};

const renderGeminiFeatures = () => {
    const placeholder = (text) => `<p>${text}</p>`;
    const loading = `<div class="flex items-center text-custom-muted-text"><i data-lucide="loader-2" class="animate-spin mr-2 h-5 w-5"></i>Generating...</div>`;

    summaryDisplay.innerHTML = state.isGenerating.summary ? loading : (state.geminiResults.summary || placeholder('Click "Generate" to create an executive summary...'));
    themesDisplay.innerHTML = state.isGenerating.themes ? loading : (state.geminiResults.themes || placeholder('Click "Analyze" to identify recurring themes...'));
    insightsDisplay.innerHTML = state.isGenerating.insights ? loading : (state.geminiResults.insights || placeholder('Ask a question to get specific insights...'));
```

```
        actionsDisplay.innerHTML = state.isGenerating.actions ? loading :
(state.geminiResults.actions || placeholder('Click "Suggest" to generate actionable
steps...'));


        generateSummaryBtn.disabled = state.texts.length === 0 ||
Object.values(state.isGenerating).some(v => v);
        analyzeThemesBtn.disabled = state.texts.length === 0 ||
Object.values(state.isGenerating).some(v => v);
        getInsightsBtn.disabled = state.texts.length === 0 ||
Object.values(state.isGenerating).some(v => v);
        suggestActionsBtn.disabled = state.texts.length === 0 ||
Object.values(state.isGenerating).some(v => v);
    };

    // --- EVENT HANDLERS ---
    const handleAnalysis = async (textToAnalyze) => {
      if (!textToAnalyze.trim()) return;
      state.isLoading = true;
      state.error = null;
      state.cancellationRequested = false;
      state.progress = 0;
      render();

      try {
        // Animate to 10% to show the process has started
        await new Promise(resolve => setTimeout(resolve, 50));
        state.progress = 10;
        renderProgress();

        const result = await callGeminiApi(textToAnalyze);
```

```
            if (!state.cancellationRequested) {
                const newText = { id: Date.now() + Math.random(), content: textToAnalyze, ...result };
                state.texts = [newText, ...state.texts];
                textInput.value = '';

                // Animate to 100% on completion
                state.progress = 100;
                renderProgress();
                await new Promise(resolve => setTimeout(resolve, 500)); // Pause to show completion
            }
        } catch (e) {
            state.error = e.message;
        } finally {
            state.isLoading = false;
            state.progress = 0; // Reset for next time
            render(); // Hide progress bar
        }
    };

    const handleBatchAnalysis = async (textsToAnalyze) => {
        if (textsToAnalyze.length === 0) return;
        state.isLoading = true;
        state.error = null;
        state.cancellationRequested = false;
        state.progress = 0;
        render();
```

```
        const results = [];
        for (let i = 0; i < textsToAnalyze.length; i++) {
            if (state.cancellationRequested) {
                console.log("Analysis cancelled by user.");
                break;
            }
            const textItem = textsToAnalyze[i];
            if (!textItem.content.trim()) {
                state.progress = ((i + 1) / textsToAnalyze.length) * 100;
                renderProgress();
                continue;
            };
            try {
                const result = await callGeminiApi(textItem.content);
                results.push({ ...result, id: Date.now() + Math.random(), content:
textItem.content, fileId: textItem.fileId });
            } catch (e) {
                console.error(`Failed to process: "${textItem.content}"`, e);
                state.error = e.message || `Partially failed. Some texts could not be
analyzed.`;
            }
            state.progress = ((i + 1) / textsToAnalyze.length) * 100;
            renderProgress();
        }
        state.texts = [...results.reverse(), ...state.texts];
        state.isLoading = false;
        render();
    };
```

```
    const handleClear = () => {
        state = { ...initialState, texts: [], uploadedFiles: [], comparisonSelection: [], geminiResults: { summary: '', themes: '', insights: '', actions: '' } };
        textInput.value = '';
        fileNameDisplay.textContent = 'No file selected';
        fileNameDisplay.title = 'No file selected';
        render();
    };

    const handleDeleteFile = (fileId) => {
        state.uploadedFiles = state.uploadedFiles.filter(f => f.id !== fileId);
        state.texts = state.texts.filter(t => t.fileId !== fileId);
        if (state.uploadedFiles.length === 0) {
            fileNameDisplay.textContent = 'No file selected';
            fileNameDisplay.title = 'No file selected';
        }
        render();
    };

    startAnalyzingBtn.addEventListener('click', () => {
        landingPage.style.display = 'none';
        dashboard.style.display = 'block';
        tourModal.style.display = 'flex';
    });

    analyzeBtn.addEventListener('click', () => handleAnalysis(textInput.value));
    cancelBtn.addEventListener('click', () => {
```

```
    state.cancellationRequested = true;
});
clearBtn.addEventListener('click', handleClear);
closeErrorBtn.addEventListener('click', () => { state.error = null; render(); });

textInput.addEventListener('input', () => {
    analyzeBtn.disabled = state.isLoading || !textInput.value.trim();
});

uploadBtn.addEventListener('click', () => {
    fileInput.click();
});

uploadedFilesList.addEventListener('click', (e) => {
    const deleteButton = e.target.closest('.delete-file-btn');
    if (deleteButton) {
        const fileId = parseFloat(deleteButton.dataset.fileId);
        handleDeleteFile(fileId);
    }
});

fileInput.addEventListener('change', (e) => {
    const files = e.target.files;
    if (files && files.length > 0) {
        if (files.length === 1) {
            fileNameDisplay.textContent = files[0].name;
            fileNameDisplay.title = files[0].name;
        } else {
```

```
            fileNameDisplay.textContent = `${files.length} files selected`;
            fileNameDisplay.title = `${files.length} files selected`;
        }

        const allChunks = [];
        const fileReadPromises = Array.from(files).map(file => {
            const fileId = Date.now() + Math.random();
            state.uploadedFiles.push({ id: fileId, name: file.name });

            return new Promise((resolve, reject) => {
                const reader = new FileReader();
                if (file.type === "application/pdf") {
                    reader.onload = function(event) {
                        const loadingTask = pdfjsLib.getDocument({ data: event.target.result
});
                        loadingTask.promise.then(function(pdf) {
                            const pagePromises = [];
                            for (let i = 1; i <= pdf.numPages; i++) {
                                pagePromises.push(
                                    pdf.getPage(i).then(page =>
page.getTextContent()).then(textContent =>
                                        textContent.items.map(item => item.str).join(' ')
                                    )
                                );
                            }
                            Promise.all(pagePromises).then(pageTexts => {
                                const allText = pageTexts.join('\n');
                                chunkText(allText).forEach(chunk => allChunks.push({ fileId,
content: chunk }));
```

```
                    resolve();
                  });
                }).catch(reject);
              };
              reader.readAsArrayBuffer(file);
            } else if (file.type === "application/vnd.openxmlformats-officedocument.wordprocessingml.document") {
              reader.onload = function(event) {
                mammoth.extractRawText({ arrayBuffer: event.target.result })
                  .then(function(result){
                    chunkText(result.value).forEach(chunk => allChunks.push({ fileId, content: chunk }));
                    resolve();
                  }).catch(reject);
              };
              reader.readAsArrayBuffer(file);
            }
            else {
              reader.onload = (event) => {
                chunkText(event.target.result).forEach(chunk => allChunks.push({ fileId, content: chunk }));
                resolve();
              };
              reader.onerror = reject;
              reader.readAsText(file);
            }
          });
        });
```

```
            Promise.all(fileReadPromises).then(() => {
                if (allChunks.length > 0) {
                    handleBatchAnalysis(allChunks);
                }
                render(); // Render the file list immediately
            }).catch(err => {
                state.error = "Error reading one or more files.";
                render();
            });
        }
        e.target.value = null;
    });

    viewDashboardBtn.addEventListener('click', () => { state.view = 'dashboard'; render(); });
    viewComparisonBtn.addEventListener('click', () => { state.view = 'comparison'; render(); });
    viewGeminiFeaturesBtn.addEventListener('click', () => { state.view = 'gemini'; render(); });

    resultsContainer.addEventListener('change', (e) => {
        if (e.target.classList.contains('comparison-checkbox')) {
            const id = parseFloat(e.target.dataset.id);
            if (e.target.checked) {
                state.comparisonSelection.push(id);
            } else {
                state.comparisonSelection = state.comparisonSelection.filter(selId => selId !== id);
            }
```

```
            comparisonCount.textContent = state.comparisonSelection.length;
        }
    });

    const handleExport = (format) => {
        if (state.texts.length === 0) return;
        if (format === 'json') {
            const dataStr = JSON.stringify(state.texts, null, 2);
            const blob = new Blob([dataStr], { type: 'application/json' });
            const url = URL.createObjectURL(blob);
            const a = document.createElement('a');
            a.href = url; a.download = 'sentiment_analysis.json'; a.click();
URL.revokeObjectURL(url);
        } else if (format === 'csv') {
            const headers = 'id,sentiment,confidence,keywords,explanation,content\n';
            const rows = state.texts.map(t =>
`${t.id},${t.sentiment},${t.confidence},"${t.keywords.join(';
')}","${t.explanation.replace(/"/g, '""')}","${t.content.replace(/"/g, '""')}"`).join('\n');
            const blob = new Blob([headers + rows], { type: 'text/csv;charset=utf-8;' });
            const url = URL.createObjectURL(blob);
            const a = document.createElement('a');
            a.href = url; a.download = 'sentiment_analysis.csv'; a.click();
URL.revokeObjectURL(url);
        } else if (format === 'pdf') {
            const { jsPDF } = window.jspdf;
            const doc = new jsPDF();
            doc.text("Sentiment Analysis Report", 14, 22);
            doc.setFontSize(10);
            doc.text(`Generated on: ${new Date().toLocaleString()}`, 14, 30);
```

```
      const bodyData = state.texts.map(t => {
        return [
          `Sentiment: ${t.sentiment} (${(t.confidence * 100).toFixed(1)}%)\n\n` +
          `Content: "${t.content}"\n\n` +
          `Explanation: ${t.explanation}\n\n` +
          `Keywords: ${t.keywords.join(', ')}`
        ];
      });

      doc.autoTable({
        startY: 40,
        body: bodyData,
        theme: 'grid',
        styles: { cellPadding: 3, fontSize: 10 },
      });

      doc.save('sentiment_analysis_report.pdf');
    }
  };

  exportBtn.addEventListener('click', (e) => {
    e.stopPropagation();
    exportMenu.classList.toggle('hidden');
  });

  document.addEventListener('click', (e) => {
    if (!exportBtn.contains(e.target) && !exportMenu.contains(e.target)) {
```

```
            exportMenu.classList.add('hidden');
        }
    });

    document.getElementById('export-csv').addEventListener('click', (e) => { e.preventDefault(); handleExport('csv'); });
    document.getElementById('export-json').addEventListener('click', (e) => { e.preventDefault(); handleExport('json'); });
    document.getElementById('export-pdf').addEventListener('click', (e) => { e.preventDefault(); handleExport('pdf'); });

    // --- GUIDED TOUR LOGIC ---
    const tourSteps = [
        {
            element: '#tour-step-1',
            title: 'Analyze Text',
            content: 'Type or paste text here, or upload files using the button. Then click "Analyze" to see the results.'
        },
        {
            element: '#tour-step-2',
            title: 'Visualize Data',
            content: 'Your results will be visualized here with gauges and charts for an at-a-glance understanding.'
        },
        {
            element: '#tour-step-3',
            title: 'Detailed Results',
            content: 'Find detailed analysis for each piece of text here, including sentiment, confidence, and keywords.'
```

```
        },
        {
            element: '#view-comparison-btn',
            title: 'Compare Texts',
            content: 'Select multiple results using the checkboxes, then click here to compare them side-by-side.'
        },
        {
            element: '#view-gemini-features-btn',
            title: '2 Point creativity features',
            content: 'After analyzing data, visit this tab to generate summaries, find themes, and get actionable insights.'
        }
    ];
    let currentTourStep = 0;

    function showTourStep(stepIndex) {
        // Clean up previous step
        const prevStep = tourSteps[stepIndex - 1];
        if (prevStep) {
            const prevElement = document.querySelector(prevStep.element);
            if (prevElement) prevElement.classList.remove('tour-highlight');
        }
        const existingTip = document.getElementById('tour-tip');
        if (existingTip) existingTip.remove();

        if (stepIndex >= tourSteps.length) {
            endTour();
            return;
```

```
    }

    const step = tourSteps[stepIndex];
    const targetElement = document.querySelector(step.element);
    if (!targetElement) {
      console.error("Tour element not found:", step.element);
      endTour();
      return;
    }

    targetElement.classList.add('tour-highlight');
    targetElement.scrollIntoView({ behavior: 'smooth', block: 'center' });

    const tip = document.createElement('div');
    tip.id = 'tour-tip';
    tip.innerHTML = `
      <h3 class="text-lg font-bold mb-2">${step.title}</h3>
      <p class="text-custom-muted-text mb-4">${step.content}</p>
      <div class="flex justify-between items-center">
        <span class="text-sm text-custom-muted-text">${stepIndex + 1} / ${tourSteps.length}</span>
        <div>
          ${stepIndex > 0 ? '<button id="tour-prev-btn" class="text-custom-muted-text font-semibold py-1 px-3 rounded-lg hover:bg-gray-700 mr-2">Prev</button>' : ''}
          <button id="tour-next-btn" class="bg-custom-primary text-white font-semibold py-1 px-4 rounded-lg hover:bg-custom-dark">${stepIndex === tourSteps.length - 1 ? 'Finish' : 'Next'}</button>
        </div>
      </div>
```

```
        `;
        document.body.appendChild(tip);

        const rect = targetElement.getBoundingClientRect();
        const tipHeight = tip.offsetHeight;
        const tipWidth = tip.offsetWidth;

        let top = rect.bottom + 10;
        let left = rect.left;

        if (top + tipHeight > window.innerHeight) {
          top = rect.top - tipHeight - 10;
        }
        if (left + tipWidth > window.innerWidth) {
          left = window.innerWidth - tipWidth - 10;
        }
        if (left < 10) left = 10;
        if (top < 10) top = 10;


        tip.style.top = `${top}px`;
        tip.style.left = `${left}px`;

        document.getElementById('tour-next-btn').onclick = () =>
showTourStep(stepIndex + 1);
        if (stepIndex > 0) {
          document.getElementById('tour-prev-btn').onclick = () =>
showTourStep(stepIndex - 1);
        }
```

```
}

function startTour() {
    tourModal.style.display = 'none';
    const overlay = document.createElement('div');
    overlay.id = 'tour-overlay';
    document.body.appendChild(overlay);
    currentTourStep = 0;
    showTourStep(0);
}

function endTour() {
    const overlay = document.getElementById('tour-overlay');
    if(overlay) overlay.remove();
    const tip = document.getElementById('tour-tip');
    if(tip) tip.remove();
    const highlighted = document.querySelector('.tour-highlight');
    if(highlighted) highlighted.classList.remove('tour-highlight');
}

tourYesBtn.addEventListener('click', startTour);
tourNoBtn.addEventListener('click', () => {
    tourModal.style.display = 'none';
});

// --- GEMINI FEATURE HANDLERS ---
async function handleGeminiFeature(featureType, prompt) {
    state.isGenerating[featureType] = true;
```

```
        render();
        try {
            const result = await generateGeminiText(prompt);
            state.geminiResults[featureType] = result.replace(/\n/g, '<br>'); // Basic formatting
        } catch (e) {
            state.error = `Failed to generate ${featureType}.`;
        } finally {
            state.isGenerating[featureType] = false;
            render();
        }
    }

    generateSummaryBtn.addEventListener('click', () => {
        const dataForPrompt = state.texts.map(t => `- (${t.sentiment}): ${t.content}`).join('\n');
        const prompt = `Based on the following sentiment data, provide a concise, high-level executive summary of the key findings in bullet points:\n\n${dataForPrompt}`;
        handleGeminiFeature('summary', prompt);
    });

    analyzeThemesBtn.addEventListener('click', () => {
        const dataForPrompt = state.texts.map(t => t.content).join('\n\n');
        const prompt = `Analyze the following texts and identify the top 5 recurring themes. For each theme, provide a brief description and classify it as primarily positive, negative, or mixed:\n\n${dataForPrompt}`;
        handleGeminiFeature('themes', prompt);
    });
```

```
        getInsightsBtn.addEventListener('click', () => {
            const question = insightsQuestion.value;
            if (!question.trim()) {
                state.error = "Please enter a question to get insights.";
                render();
                return;
            }
            const dataForPrompt = state.texts.map(t => `- (${t.sentiment}): ${t.content}`).join('\n');
            const prompt = `Given the following sentiment data, answer this question: "${question}"\n\nData:\n${dataForPrompt}`;
            handleGeminiFeature('insights', prompt);
        });

        suggestActionsBtn.addEventListener('click', () => {
            const dataForPrompt = state.texts.map(t => `- (${t.sentiment}): ${t.content}`).join('\n');
            const prompt = `Based on the following sentiment analysis, suggest three actionable business steps to improve customer satisfaction or product offerings. Format the response as a numbered list:\n\n${dataForPrompt}`;
            handleGeminiFeature('actions', prompt);
        });


        // --- INITIAL RENDER ---
        render();
    });
  </script>
</body>
```

</html>