This is +page.svelte “<script>

```
  import { onMount } from 'svelte';
  import { chart } from '$lib/chartAction.js';
  import { filterStore, activeFilterCount } from '$lib/stores/filterStore.js';
  import Map from '$lib/Map.svelte';
  import Heatmap from '$lib/Heatmap.svelte';
  import NuanceCards from '$lib/NuanceCards.svelte';
  import ImpactfulComments from '$lib/ImpactfulComments.svelte';
  import SentimentChart from '$lib/components/SentimentChart.svelte';
 import { forceCenter, randomWeibull } from 'd3';


  let drafts = [];
  let allComments = [];
  let sections = [];
  let selectedDraftId = '';
  let selectedSectionId = 'all';
  let selectedState = 'All';
  let selectedAction = 'All';
  let selectedSentiment = 'All';


  let totalComments = 0;
  let positiveCount = 0;
  let neutralCount = 0;
  let negativeCount = 0;
  let sentimentScore = 0;
```

```
let gaugeChartConfig = {};
let sentimentBarChartConfig = {};
let actionBarChartConfig = {};
let stateChartConfig = {};
let industryChartConfig = {};
let sectionChartConfig = {};

const API_BASE_URL = 'http://127.0.0.1:5000';

// Subscribe to filter store
let currentFilters = {};
filterStore.subscribe(value => {
    currentFilters = value;
});

// Apply filters from store to commentsForDisplay
$: commentsForDisplay = (() => {
    if (!selectedDraftId) return [];
    let filtered = allComments;

    // Apply cross-filter store filters
    if (currentFilters.section) {
        const section = sections.find(s => s.section_title === currentFilters.section);
        if (section) {
            filtered = filtered.filter(c => c.section_id == section.section_id);
        }
```

```
}
if (currentFilters.state) {
    filtered = filtered.filter(c => c.state == currentFilters.state);
}
if (currentFilters.action) {
    filtered = filtered.filter(c => c.action_type == currentFilters.action);
}
if (currentFilters.sentiment) {
    filtered = filtered.filter(c => c.sentiment_label == currentFilters.sentiment);
}
if (currentFilters.industry) {
    filtered = filtered.filter(c => {
        const ind = c.industry || 'Individual';
        return ind === currentFilters.industry;
    });
}

// Apply dropdown filters (if not using cross-filter)
if (selectedSectionId !== 'all' && !currentFilters.section) {
    filtered = filtered.filter(c => c.section_id == selectedSectionId);
}
if (selectedState !== 'All' && !currentFilters.state) {
    filtered = filtered.filter(c => c.state == selectedState);
}
if (selectedAction !== 'All' && !currentFilters.action) {
    filtered = filtered.filter(c => c.action_type == selectedAction);
```

```
    }
    if (selectedSentiment !== 'All' && !currentFilters.sentiment) {
      filtered = filtered.filter(c => c.sentiment_label == selectedSentiment);
    }

    return filtered;
  })();
  // compute average scores across currently filtered selection (so SentimentChart can use them)
  $: avg_score_positive = commentsForDisplay && commentsForDisplay.length
    ? commentsForDisplay.reduce((sum, c) => sum + (c.score_positive || 0), 0) / commentsForDisplay.length
    : 0;

  $: avg_score_negative = commentsForDisplay && commentsForDisplay.length
    ? commentsForDisplay.reduce((sum, c) => sum + (c.score_negative || 0), 0) / commentsForDisplay.length
    : 0;

  $: avg_score_neutral = commentsForDisplay && commentsForDisplay.length
    ? commentsForDisplay.reduce((sum, c) => sum + (c.score_neutral || 0), 0) / commentsForDisplay.length
    : 0;

  // Calculate metrics and update charts
  $: {
    if (!commentsForDisplay || commentsForDisplay.length === 0) {
```

```
    sentimentScore = 0;
    totalComments = 0;
    positiveCount = 0;
    negativeCount = 0;
    neutralCount = 0;
} else {
    totalComments = commentsForDisplay.length;
    positiveCount = commentsForDisplay.filter(c => c.sentiment_label === 'Positive').length;
    neutralCount = commentsForDisplay.filter(c => c.sentiment_label === 'Neutral').length;
    negativeCount = commentsForDisplay.filter(c => c.sentiment_label === 'Negative').length;
    sentimentScore = (positiveCount / totalComments) * 100;
}

const actionCounts = {
    'Suggest removal': 0,
    'In Agreement': 0,
    'Suggest modification': 0
};
commentsForDisplay.forEach(c => {
    if (actionCounts.hasOwnProperty(c.action_type)) {
        actionCounts[c.action_type]++;
    }
});

const stateData = {};
commentsForDisplay.forEach(c => {
```

```
    if (!c.state) return;
    if (!stateData[c.state]) {
        stateData[c.state] = { Positive: 0, Neutral: 0, Negative: 0 };
    }
    if (c.sentiment_label) {
        stateData[c.state][c.sentiment_label]++;
    }
});

const industryData = {};
commentsForDisplay.forEach(c => {
    const industry = c.industry || 'Individual';
    if (!industryData[industry]) {
        industryData[industry] = { Positive: 0, Neutral: 0, Negative: 0 };
    }
    if (c.sentiment_label) {
        industryData[industry][c.sentiment_label]++;
    }
});

const sectionData = {};
commentsForDisplay.forEach(c => {
    if (!c.section_title) return;
    const shortTitle = c.section_title.length > 30
        ? c.section_title.substring(0, 30) + '...'
        : c.section_title;
```

```
    if (!sectionData[shortTitle]) {
        sectionData[shortTitle] = { Positive: 0, Neutral: 0, Negative: 0 };
    }
    if (c.sentiment_label) {
        sectionData[shortTitle][c.sentiment_label]++;
    }
});

const normalizedScore = sentimentScore + 100;
const sentimentColor = getSentimentColor(sentimentScore);

gaugeChartConfig = {
    type: 'doughnut',
    data: {
        datasets: [{
            data: [normalizedScore, 200 - normalizedScore],
            backgroundColor: [sentimentColor, '#f0f0f0'],
            borderWidth: 0,
            circumference: 180,
            rotation: 270,
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        cutout: '75%',
```

```
        plugins: {
            tooltip: { enabled: false },
            legend: { display: false },
            datalabels: { display: false }
        }
    }
};

sentimentBarChartConfig = {
    type: 'bar',
    data: {
        labels: ['Positive', 'Neutral', 'Negative'],
        datasets: [{
            data: [positiveCount, neutralCount, negativeCount],
            backgroundColor: ['#28a745', '#007bff', '#dc3545'],
            borderRadius: 6,
            barThickness: 50
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        onClick: (event, elements) => {
            if (elements.length > 0) {
                const index = elements[0].index;
                const sentiments = ['Positive', 'Neutral', 'Negative'];
```

```
        filterStore.setFilter('sentiment', sentiments[index]);
      }
    },
    plugins: {
      legend: { display: false },
      tooltip: {
        backgroundColor: '#fff',
        titleColor: '#333',
        bodyColor: '#666',
        borderColor: '#ddd',
        borderWidth: 1,
        padding: 10
      },
      datalabels: {
        anchor: 'end',
        align: 'top',
        color: '#333',
        font: { weight: 'bold', size: 13 }
      }
    },
    scales: {
      y: {
        beginAtZero: true,
        ticks: { stepSize: 1, color: '#666', font: { size: 11 } },
        grid: { color: '#e9ecef', drawBorder: false }
      },
```

```
            x: {
                grid: { display: false },
                ticks: { color: '#333', font: { size: 12, weight: '600' } }
            }
        }
    };

    actionBarChartConfig = {
        type: 'pie',
        data: {
            labels: ['Suggest\nremoval', 'In Agreement', 'Suggest\nmodification'],
            datasets: [{
                data: [actionCounts['Suggest removal'], actionCounts['In Agreement'],
actionCounts['Suggest modification']],
                backgroundColor: ['#dc3545', '#28a745', '#007bff']
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
            onClick: (event, elements) => {
                if (elements.length > 0) {
                    const index = elements[0].index;
                    const actions = ['Suggest removal', 'In Agreement', 'Suggest modification'];
                    filterStore.setFilter('action', actions[index]);
```

```
    }
  },
  plugins: {
    legend: {
      display: true,
      position: 'right',
      labels: {
        usePointStyle: true,
        padding: 12,
        font: { size: 11 }
      }
    },
    tooltip: {
      backgroundColor: '#fff',
      titleColor: '#333',
      bodyColor: '#666',
      borderColor: '#ddd',
      borderWidth: 1,
      padding: 10
    },
    datalabels: {
      color: '#fff',
      formatter: (value, context) => {
        let sum = context.chart.data.datasets[0].data.reduce((a, b) => a + b, 0);
        return sum > 0 ? ((value * 100 / sum).toFixed(1) + "%") : '';
      },
```

```
            font: { weight: 'bold', size: 13 }
        }
      }
    }
};

const stateLabels = Object.keys(stateData);
stateChartConfig = {
    type: 'bar',
    data: {
      labels: stateLabels,
      datasets: [
        {
          label: 'Negative',
          data: stateLabels.map(s => stateData[s].Negative),
          backgroundColor: '#dc3545'
        },
        {
          label: 'Neutral',
          data: stateLabels.map(s => stateData[s].Neutral),
          backgroundColor: '#007bff'
        },
        {
          label: 'Positive',
          data: stateLabels.map(s => stateData[s].Positive),
          backgroundColor: '#28a745'
```

```
        }
      ]
    },
    options: {
      indexAxis: 'y',
      responsive: true,
      maintainAspectRatio: false,
      onClick: (event, elements) => {
        if (elements.length > 0) {
          const index = elements[0].index;
          filterStore.setFilter('state', stateLabels[index]);
        }
      },
      plugins: {
        legend: {
          display: true,
          position: 'top',
          labels: { usePointStyle: true, padding: 12, font: { size: 11 } }
        },
        tooltip: {
          backgroundColor: '#fff',
          titleColor: '#333',
          bodyColor: '#666',
          borderColor: '#ddd',
          borderWidth: 1
        },
```

```
            datalabels: {
                color: '#fff',
                font: { weight: 'bold', size: 10 },
                formatter: (value) => value > 0 ? value : ''
            }
        },
        scales: {
            x: {
                stacked: true,
                beginAtZero: true,
                grid: { color: '#e9ecef' },
                ticks: { color: '#666', font: { size: 10 } }
            },
            y: {
                stacked: true,
                grid: { display: false },
                ticks: { color: '#333', font: { size: 11 } }
            }
        }
    }
};

const industryLabels = Object.keys(industryData);
industryChartConfig = {
    type: 'bar',
    data: {
```

```
    labels: industryLabels,
    datasets: [
      {
        label: 'Negative',
        data: industryLabels.map(i => industryData[i].Negative),
        backgroundColor: '#dc3545'
      },
      {
        label: 'Neutral',
        data: industryLabels.map(i => industryData[i].Neutral),
        backgroundColor: '#007bff'
      },
      {
        label: 'Positive',
        data: industryLabels.map(i => industryData[i].Positive),
        backgroundColor: '#28a745'
      }
    ]
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    onClick: (event, elements) => {
      if (elements.length > 0) {
        const index = elements[0].index;
        filterStore.setFilter('industry', industryLabels[index]);
```

```
    }
  },
  plugins: {
    legend: {
      display: true,
      position: 'top',
      labels: { usePointStyle: true, padding: 12, font: { size: 11 } }
    },
    tooltip: {
      backgroundColor: '#fff',
      titleColor: '#333',
      bodyColor: '#666',
      borderColor: '#ddd',
      borderWidth: 1
    },
    datalabels: {
      anchor: 'end',
      align: 'top',
      color: '#333',
      font: { weight: 'bold', size: 10 },
      formatter: (value) => value > 0 ? value : ''
    }
  },
  scales: {
    y: {
      beginAtZero: true,
```

```
            ticks: { stepSize: 1, color: '#666', font: { size: 10 } },
            grid: { color: '#e9ecef' }
          },
          x: {
            grid: { display: false },
            ticks: { color: '#333', font: { size: 10 } }
          }
        }
    }
};

const sectionLabels = Object.keys(sectionData);
sectionChartConfig = {
    type: 'bar',
    data: {
        labels: sectionLabels,
        datasets: [
            {
                label: 'Negative',
                data: sectionLabels.map(s => sectionData[s].Negative),
                backgroundColor: '#dc3545'
            },
            {
                label: 'Neutral',
                data: sectionLabels.map(s => sectionData[s].Neutral),
                backgroundColor: '#007bff'
```

```
      },
      {
        label: 'Positive',
        data: sectionLabels.map(s => sectionData[s].Positive),
        backgroundColor: '#28a745'
      }
    ]
  },
  options: {
    indexAxis: 'y',
    responsive: true,
    maintainAspectRatio: false,
    onClick: (event, elements) => {
      if (elements.length > 0) {
        const index = elements[0].index;
        const fullTitle = sections.find(s =>
          s.section_title.startsWith(sectionLabels[index].replace('...', ''))
        )?.section_title || sectionLabels[index];
        filterStore.setFilter('section', fullTitle);
      }
    },
    plugins: {
      legend: {
        display: true,
        position: 'top',
        labels: { usePointStyle: true, padding: 12, font: { size: 11 } }
```

```
    },
    tooltip: {
      backgroundColor: '#fff',
      titleColor: '#333',
      bodyColor: '#666',
      borderColor: '#ddd',
      borderWidth: 1
    },
    datalabels: {
      anchor: 'end',
      align: 'left',
      color: 'white',
      font: { weight: 'bold', size: 12 },
      formatter: (value) => value > 0 ? value : ''
    }
  },
  scales: {
    y: {
      beginAtZero: true,
      ticks: { stepSize: 1, color: '#666', font: { size: 10 } },
      grid: { color: '#e9ecef' },
      stacked: true
    },
    x: {
      grid: { display: false },
      ticks: {
```

```
                color: '#333',
                font: { size: 9 },
                maxRotation: 45,
                minRotation: 45
            },
            stacked: true
          }
        }
      }
    };
}

$: selectedDraft = drafts.find(d => d.draft_id == selectedDraftId) || null;
$: availableStates = ['All', ...new Set(allComments.map(c => c.state).filter(c => c))];

$: stateMapData = (() => {
    const data = {};
    commentsForDisplay.forEach(c => {
      if (!c.state) return;
      if (!data[c.state]) {
        data[c.state] = { positive: 0, neutral: 0, negative: 0, total: 0 };
      }
      data[c.state].total++;
      if (c.sentiment_label === 'Positive') data[c.state].positive++;
      if (c.sentiment_label === 'Neutral') data[c.state].neutral++;
      if (c.sentiment_label === 'Negative') data[c.state].negative++;
```

```
    });
    return data;
})();

onMount(async () => {
    try {
        const res = await fetch(`${API_BASE_URL}/api/drafts`);
        drafts = await res.json();
    } catch (error) {
        console.error('Error fetching drafts:', error);
    }
});

$: if (selectedDraftId) handleDraftChange();

async function handleDraftChange() {
if (!selectedDraftId) {
    sections = [];
    allComments = [];
    return;
}
selectedSectionId = 'all';
selectedState = 'All';
selectedAction = 'All';
selectedSentiment = 'All';
filterStore.reset();
```

```
    try {
        const [commentsRes, sectionsRes] = await Promise.all([
            fetch(`${API_BASE_URL}/api/comments/${selectedDraftId}`),
            fetch(`${API_BASE_URL}/api/sections/${selectedDraftId}`)
        ]);
         allComments = await commentsRes.json();
    sections = await sectionsRes.json();
        allComments = allComments.filter(comment =>
        sections.some(section => section.section_id === comment.section_id)
    );
    } catch (error) {
        console.error('Error fetching data:', error);
    }
}

    function getSentimentColor(score) {
        if (score < 40) return '#dc3545';
        if (score < 60) return '#ffc107';
        return '#28a745';
    }

    function getSentimentText(score) {
        if (score < 40) return 'Negative';
        if (score < 60) return 'Neutral';
        return 'Positive';
```

```
  }
</script>

<svelte:head>
  <title>Public Consultation Analysis Dashboard</title>
</svelte:head>

<div class="container">
  <div class="controls">
    <div class="control-group">
      <label for="draft-select">Draft Selection</label>
      <select id="draft-select" bind:value={selectedDraftId}>
        <option value="">-- Select a Draft --</option>
        {#each drafts as draft}
          <option value={draft.draft_id}>{draft.title}</option>
        {/each}
      </select>
    </div>

<style>
  :global(body) {
    margin: 0;
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
    background: #f5f5f5;
  }
```

```
.container {
    height: 100vh;
    width:100%;
    margin: 0 auto;
}

.controls {
    display: flex;
    gap: 1rem;
    margin-bottom: 1.5rem;
    flex-wrap: nowrap;
    align-items: center;
    overflow-x: auto;
}

.control-group {
    display: flex;
    flex-direction: column;
    gap: 0.2rem;
    width: 250px;
    flex-shrink: 0;
}

.control-group label {
    font-size: 0.85rem;
    font-weight: 600;
```

```
    color: #495057;
}

.filter-badges {
    display: flex;
    gap: 0.5rem;
    margin-bottom: 1rem;
    flex-wrap: wrap;
    align-items: center;
    padding: 0.75rem;
    background: #e7f3ff;
    border-radius: 8px;
    border-left: 4px solid #0d6efd;
}

.badge-label {
    font-size: 0.85rem;
    font-weight: 600;
    color: #0d6efd;
}

.filter-badge {
    display: inline-flex;
    align-items: center;
    gap: 0.3rem;
    padding: 0.4rem 0.7rem;
```

```
    background: #0d6efd;
    color: white;
    border: none;
    border-radius: 16px;
    font-size: 0.8rem;
    font-weight: 500;
    cursor: pointer;
    transition: all 0.2s;
}

.filter-badge:hover {
    background: #0b5ed7;
    transform: scale(1.05);
}

.clear-all-btn {
    padding: 0.4rem 0.9rem;
    background: #dc3545;
    color: white;
    border: none;
    border-radius: 16px;
    font-size: 0.8rem;
    font-weight: 600;
    cursor: pointer;
    transition: all 0.2s;
    margin-left: auto;
```

```
}

.clear-all-btn:hover {
    background: #bb2d3b;
    transform: scale(1.05);
}

.comment-summary {
    display: flex;
    justify-content: space-around;
    /* margin-bottom: 1.5rem; */
    flex-wrap: wrap;
}

.comment-box {
    background: white;
    border-radius: 10px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.08);
    text-align: center;
    padding:0.5rem 1rem;
    display:flex;
    justify-content:center;
    align-items:center;
    flex-direction:column;
    max-height:85px;
}
```

```
.comment-box h3 {
    font-size: 0.9rem;
    font-weight: 600;
    color: #495057;
    margin: 0;
    /* margin-bottom: 0.5rem; */
}

.comment-box p {
    font-size: 1.2rem;
    font-weight: bold;
    color: #343a40;
    margin: 0;
}

.comment-box.total {
    border-bottom: 3px solid #6c757d;
}

.comment-box.positive {
    border-bottom: 3px solid #28a745;
}

.comment-box.neutral {
    border-bottom: 3px solid #007bff;
```

```
}

.comment-box.negative {
    border-bottom: 3px solid #dc3545;
}

select {
    font-size: 0.9rem;
    padding: 0.5rem 0.9rem;
    border-radius: 6px;
    border: 2px solid #dee2e6;
    background: white;
    cursor: pointer;
    width: 100%;
    transition: border-color 0.2s;
    box-sizing: border-box;
}

select:hover {
    border-color: #adb5bd;
}

select:focus {
    outline: none;
    border-color: #0d6efd;
}
```

```
.dashboard-grid {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: auto auto auto auto auto;
    gap: 1rem;
}

.gauge-card {
    grid-column: 1 / 2;
    grid-row: 1;
}

.action-bar-card {
    grid-column: 2 / 3;
    grid-row: 1;
}

.sentiment-bar-card {
    grid-column: 3 / 4;
    grid-row: 3;
}

.map-card {
    grid-column: 3 / 4;
    grid-row: 1 / 3;
```

```
}

.industry-card {
    grid-column: 1 / 3;
    grid-row: 3;
}

.section-card {
    grid-column: 1 / 3;
    grid-row: 2;
    display: flex;
    justify-content: center;
    align-items: center;
}

.heatmap-card {
    grid-column: 1/ 4;
    grid-row: 4;
}

.impactful-card {
    grid-column: 1 / 4;
    grid-row: 5;
}

.impactful-wrapper {
```

```
    padding: 1.2rem;
}

.card {
    background: white;
    border-radius: 10px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.08);
    display: flex;
    flex-direction: column;
    overflow: hidden;
    transition: box-shadow 0.3s;
}

.clickable-chart {
    cursor: pointer;
}

.clickable-chart:hover {
    box-shadow: 0 4px 12px rgba(13, 110, 253, 0.2);
}

.card h3 {
    margin: 0;
    padding: 1rem;
    text-align: center;
    font-size: 0.9rem;
```

```
    font-weight: 600;
    color: #495057;
    border-bottom: 2px solid #f1f3f5;
    background: #fafafa;
}

.click-hint {
    font-size: 0.7rem;
    color: #0d6efd;
    font-weight: 500;
    font-style: italic;
}

.gauge-wrapper {
    padding: 1.5rem 1rem;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.gauge-canvas-container {
    position: relative;
    width: 100%;
    max-width: 250px;
    height: 125px;
}
```

```
.gauge-labels {
    position: relative;
    width: 100%;
    max-width: 250px;
    margin-top: 0.3rem;
}

.gauge-label {
    position: absolute;
    font-size: 0.7rem;
    color: #6c757d;
    font-weight: 500;
}

.min-label {
    left: 0;
}

.max-label {
    right: 0;
}

.gauge-center {
    text-align: center;
    margin-top: 0.8rem;
```

```
}

.gauge-value {
    font-size: 1.8rem;
    font-weight: 700;
    margin-bottom: 0.2rem;
}

.gauge-text {
    font-size: 1.1rem;
    font-weight: 600;
}

.chart-wrapper {
    padding: 1.2rem;
    height: 250px;
    width:80%;
}

.map-container {
    padding: 1.2rem;
    position: relative;
    height: 100%;
    display: flex;
    flex-direction: column;
}
```

```
.map-legend {
    display: flex;
    justify-content: center;
    gap: 1.5rem;
    margin-top: auto;
    padding-top: 0.8rem;
    border-top: 1px solid #e9ecef;
}

.legend-item {
    display: flex;
    align-items: center;
    gap: 0.4rem;
}

.legend-color {
    width: 16px;
    height: 16px;
    border-radius: 50%;
    border: 2px solid #fff;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
}

.legend-item span {
    font-size: 0.8rem;
```

```
    color: #495057;
    font-weight: 500;
  }

  .placeholder {
    text-align: center;
    padding: 3rem 2rem;
    color: #6c757d;
    font-size: 1rem;
  }
</style>

    {#if selectedDraftId}
      <div class="control-group">
        <label for="section-select">Select Section</label>
        <select id="section-select" bind:value={selectedSectionId}>
          <option value="all">All</option>
          {#each sections as section}
            <option value={section.section_id}>{section.section_title}</option>
          {/each}
        </select>
      </div>
      <div class="control-group">
        <label for="state-select">Select a State to analyze</label>
        <select id="state-select" bind:value={selectedState}>
          {#each availableStates as state}
```

```
                <option value={state}>{state}</option>
              {/each}
            </select>
          </div>
          <div class="control-group">
            <label for="action-select">Select Action</label>
            <select id="action-select" bind:value={selectedAction}>
              <option value="All">All</option>
              <option value="Suggest removal">Suggest removal</option>
              <option value="In Agreement">In Agreement</option>
              <option value="Suggest modification">Suggest modification</option>
            </select>
          </div>
          <div class="control-group">
            <label for="sentiment-select">Select Sentiment</label>
            <select id="sentiment-select" bind:value={selectedSentiment}>
              <option value="All">All</option>
              <option value="Positive">Positive</option>
              <option value="Neutral">Neutral</option>
              <option value="Negative">Negative</option>
            </select>
          </div>
        {/if}
      </div>

      {#if selectedDraftId && $activeFilterCount > 0}
```

```
<div class="filter-badges">
  <span class="badge-label">Active Filters ({$activeFilterCount}):</span>
  {#if currentFilters.section}
    <button class="filter-badge" on:click={() => filterStore.clearFilter('section')}>
      Section: {currentFilters.section} ×
    </button>
  {/if}
  {#if currentFilters.state}
    <button class="filter-badge" on:click={() => filterStore.clearFilter('state')}>
      State: {currentFilters.state} ×
    </button>
  {/if}
  {#if currentFilters.action}
    <button class="filter-badge" on:click={() => filterStore.clearFilter('action')}>
      Action: {currentFilters.action} ×
    </button>
  {/if}
  {#if currentFilters.sentiment}
    <button class="filter-badge" on:click={() => filterStore.clearFilter('sentiment')}>
      Sentiment: {currentFilters.sentiment} ×
    </button>
  {/if}
  {#if currentFilters.industry}
    <button class="filter-badge" on:click={() => filterStore.clearFilter('industry')}>
      Industry: {currentFilters.industry} ×
    </button>
```

```
        {/if}
        <button class="clear-all-btn" on:click={() => filterStore.clearAllFilters()}>
          Clear All
        </button>
    </div>
{/if}

{#if selectedDraftId}
<div class="comment-summary">
    <div class="comment-box total">
        <h3>Total Comments</h3>
        <p>{totalComments}</p>
    </div>
    <div class="comment-box positive">
        <h3>Positive Comments</h3>
        <p>{positiveCount}</p>
    </div>
    <div class="comment-box neutral">
        <h3>Neutral Comments</h3>
        <p>{neutralCount}</p>
    </div>
    <div class="comment-box negative">
        <h3>Negative Comments</h3>
        <p>{negativeCount}</p>
    </div>
    {#if selectedDraftId}
```

```
   <NuanceCards {commentsForDisplay} />
{/if}
   </div>
    {/if}


   {#if selectedDraft}
      <div class="dashboard-grid">
         <div class="card gauge-card clickable-chart">
  <h3>Overall Sentiment Distribution <span class="click-hint">Click to filter</span></h3>
  <SentimentChart {commentsForDisplay} />
</div>


         <div class="card action-bar-card clickable-chart">
            <h3>No of Comments received in each Action <span class="click-hint">Click to filter</span></h3>
            <div class="chart-wrapper">
               <canvas use:chart={actionBarChartConfig}></canvas>
            </div>
         </div>

         <div class="card sentiment-bar-card clickable-chart">
            <h3>No of Comments received in each Sentimental Label <span class="click-hint">Click to filter</span></h3>
            <div class="chart-wrapper">
               <canvas use:chart={sentimentBarChartConfig}></canvas>
```

```
    </div>
  </div>

  <div class="card section-card clickable-chart">
    <h3>Section Wise Sentiment Analysis of Comments <span class="click-hint">Click to filter</span></h3>
    <div class="chart-wrapper">
      <canvas use:chart={sectionChartConfig}></canvas>
    </div>
  </div>

  <div class="card map-card">
    <h3>Geographic Distribution of Sentiment</h3>
    <div class="map-container">
      <Map {stateMapData} {selectedDraftId} fetchFromApi={false} on:stateClick={(e) => filterStore.setFilter('state', e.detail)} />
      <div class="map-legend">
        <div class="legend-item">
          <div class="legend-color" style="background: #28a745;"></div>
          <span>Positive</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background: #007bff;"></div>
          <span>Neutral</span>
        </div>
        <div class="legend-item">
          <div class="legend-color" style="background: #dc3545;"></div>
```

```
                <span>Negative</span>
              </div>
            </div>
          </div>
        </div>

        <div class="card industry-card clickable-chart">
          <h3>No of comments Received from Stake Holders with Sentiment <span class="click-hint">Click to filter</span></h3>
          <div class="chart-wrapper">
            <canvas use:chart={industryChartConfig}></canvas>
          </div>
        </div>
         <div class="card heatmap-card">
        <h3>Industry × Section Sentiment Heatmap</h3>
        <div class="chart-wrapper" style="height: auto;">
          <Heatmap {commentsForDisplay} on:cellClick={(e) => {
            filterStore.setFilter('industry', e.detail.industry);
            filterStore.setFilter('section', e.detail.section);
          }} />
        </div>
      </div>

      <div class="card impactful-card">
        <h3>Most Impactful Comments</h3>
        <div class="impactful-wrapper">
```

```
            <ImpactfulComments {commentsForDisplay} />
          </div>
        </div>
        </div>
    {:else}
        <div class="placeholder">
          <p>Please select a draft to view analysis</p>
        </div>
    {/if}
    </div>
```

“ and this is overall-analysis +page.svelte “

```
<script>
    import { onMount } from 'svelte';
    import { chart } from '$lib/chartAction.js';
    import Map from '$lib/Map.svelte';
    import Heatmap from '$lib/Heatmap.svelte';
    import SentimentChart from '$lib/components/SentimentChart.svelte';
    import NuanceCards from '$lib/NuanceCards.svelte';

    // --- State Management ---
    let drafts = [];
    let allComments = [];
    let sections = [];
    let selectedDraftId = '';
    let selectedSectionId = 'all';
    let selectedState = 'All';
    let selectedAction = 'All';
```

```
let selectedSentiment = 'All';

// Metrics
let totalComments = 0;
let positiveCount = 0;
let neutralCount = 0;
let negativeCount = 0;
let sentimentScore = 0;

// Chart configs
let gaugeChartConfig = {};
let sentimentBarChartConfig = {};
let actionPieChartConfig = {};
let sectionPieChartConfig = {};

// --- API Configuration ---
const API_BASE_URL = 'http://127.0.0.1:5000';

// --- Derived State ---
$: commentsForDisplay = (() => {
    if (!selectedDraftId) return [];
    let filtered = allComments;

    if (selectedSectionId !== 'all') {
        filtered = filtered.filter(c => c.section_id == selectedSectionId);
    }
```

```
    if (selectedState !== 'All') {
        filtered = filtered.filter(c => c.state == selectedState);
    }
    if (selectedAction !== 'All') {
        filtered = filtered.filter(c => c.action_type == selectedAction);
    }
    if (selectedSentiment !== 'All') {
        filtered = filtered.filter(c => c.sentiment_label == selectedSentiment);
    }

    return filtered;
})();

$: selectedDraft = drafts.find(d => d.draft_id == selectedDraftId) || null;
$: availableStates = ['All', ...new Set(allComments.map(c => c.state).filter(c => c))];

$: stateMapData = (() => {
    const data = {};
    commentsForDisplay.forEach(c => {
        if (!c.state) return;
        if (!data[c.state]) {
            data[c.state] = { positive: 0, neutral: 0, negative: 0, total: 0 };
        }
        data[c.state].total++;
        if (c.sentiment_label === 'Positive') data[c.state].positive++;
        if (c.sentiment_label === 'Neutral') data[c.state].neutral++;
```

```
        if (c.sentiment_label === 'Negative') data[c.state].negative++;
    });
    return data;
})();

// Calculate metrics and update charts
$: {
    if (!commentsForDisplay || commentsForDisplay.length === 0) {
        sentimentScore = 0;
        totalComments = 0;
        positiveCount = 0;
        negativeCount = 0;
        neutralCount = 0;
    } else {
        totalComments = commentsForDisplay.length;
        positiveCount = commentsForDisplay.filter(c => c.sentiment_label === 'Positive').length;
        neutralCount = commentsForDisplay.filter(c => c.sentiment_label === 'Neutral').length;
        negativeCount = commentsForDisplay.filter(c => c.sentiment_label === 'Negative').length;
        sentimentScore = (positiveCount / totalComments) * 100;
    }

    // Action type counts
    const actionCounts = {
        'Suggest removal': 0,
        'In Agreement': 0,
        'Suggest modification': 0
```

```
};
commentsForDisplay.forEach(c => {
   if (actionCounts.hasOwnProperty(c.action_type)) {
      actionCounts[c.action_type]++;
   }
});

// Section-wise comment counts
const sectionCounts = {};
commentsForDisplay.forEach(c => {
   if (!c.section_title) return;
   const shortTitle = c.section_title.length > 30
      ? c.section_title.substring(0, 30) + '...'
      : c.section_title;
   sectionCounts[shortTitle] = (sectionCounts[shortTitle] || 0) + 1;
});

const normalizedScore = sentimentScore + 100;
const sentimentColor = getSentimentColor(sentimentScore);

// Gauge Chart
gaugeChartConfig = {
   type: 'doughnut',
   data: {
      datasets: [{
         data: [normalizedScore, 200 - normalizedScore],
```

```
            backgroundColor: [sentimentColor, '#f0f0f0'],
            borderWidth: 0,
            circumference: 180,
            rotation: 270,
        }]
    },
    options: {
        responsive: true,
        maintainAspectRatio: false,
        cutout: '75%',
        plugins: {
            tooltip: { enabled: false },
            legend: { display: false },
            datalabels: { display: false }
        }
    }
};

// Sentiment Bar Chart
sentimentBarChartConfig = {
    type: 'bar',
    data: {
        labels: ['Positive', 'Neutral', 'Negative'],
        datasets: [{
            data: [positiveCount, neutralCount, negativeCount],
            backgroundColor: ['#28a745', '#007bff', '#dc3545'],
```

```
        borderRadius: 6,
        barThickness: 50
    }]
},
options: {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
        legend: { display: false },
        tooltip: {
            backgroundColor: '#fff',
            titleColor: '#333',
            bodyColor: '#666',
            borderColor: '#ddd',
            borderWidth: 1,
            padding: 10
        },
        datalabels: {
            anchor: 'end',
            align: 'top',
            color: '#333',
            font: { weight: 'bold', size: 13 }
        }
    },
    scales: {
        y: {
```

```
                beginAtZero: true,
                ticks: { stepSize: 1, color: '#666', font: { size: 11 } },
                grid: { color: '#e9ecef', drawBorder: false }
            },
            x: {
                grid: { display: false },
                ticks: { color: '#333', font: { size: 12, weight: '600' } }
            }
        }
    };

    // Action Type Pie Chart
    actionPieChartConfig = {
        type: 'pie',
        data: {
            labels: ['Suggest removal', 'In Agreement', 'Suggest modification'],
            datasets: [{
                data: [actionCounts['Suggest removal'], actionCounts['In Agreement'],
actionCounts['Suggest modification']],
                backgroundColor: ['#dc3545', '#28a745', '#007bff']
            }]
        },
        options: {
            responsive: true,
            maintainAspectRatio: false,
```

```
    plugins: {
      legend: {
        display: true,
        position: 'right',
        labels: { usePointStyle: true, padding: 12, font: { size: 11 } }
      },
      tooltip: {
        backgroundColor: '#fff',
        titleColor: '#333',
        bodyColor: '#666',
        borderColor: '#ddd',
        borderWidth: 1,
        padding: 10
      },
      datalabels: {
        color: '#fff',
        formatter: (value, context) => {
          let sum = context.chart.data.datasets[0].data.reduce((a, b) => a + b, 0);
          return sum > 0 ? ((value * 100 / sum).toFixed(1) + "%") : '';
        },
        font: { weight: 'bold', size: 13 }
      }
    }
  }
};
```

```
// Section Pie Chart
const sectionLabels = Object.keys(sectionCounts);
const sectionData = Object.values(sectionCounts);

sectionPieChartConfig = {
  type: 'pie',
  data: {
    labels: sectionLabels,
    datasets: [{
      data: sectionData,
      backgroundColor: [
        '#FF6384', '#36A2EB', '#FFCE56', '#4BC0C0', '#9966FF',
        '#FF9F40', '#FF6384', '#C9CBCF', '#4BC0C0', '#FF6384'
      ]
    }]
  },
  options: {
    responsive: true,
    maintainAspectRatio: false,
    plugins: {
      legend: {
        display: true,
        position: 'right',
        labels: { usePointStyle: true, padding: 8, font: { size: 10 } }
      },
      tooltip: {
```

```
            backgroundColor: '#fff',
            titleColor: '#333',
            bodyColor: '#666',
            borderColor: '#ddd',
            borderWidth: 1,
            padding: 10
          },
          datalabels: {
            color: '#fff',
            formatter: (value, context) => {
              let sum = context.chart.data.datasets[0].data.reduce((a, b) => a + b, 0);
              return sum > 0 ? ((value * 100 / sum).toFixed(1) + "%") : '';
            },
            font: { weight: 'bold', size: 11 }
          }
        }
      }
    };
}

onMount(async () => {
  try {
    const res = await fetch(`${API_BASE_URL}/api/drafts`);
    drafts = await res.json();
  } catch (error) {
    console.error('Error fetching drafts:', error);
```

```
  }
});

$: if (selectedDraftId) handleDraftChange();

async function handleDraftChange() {
  if (!selectedDraftId) {
    sections = [];
    allComments = [];
    return;
  }

  selectedSectionId = 'all';
  selectedState = 'All';
  selectedAction = 'All';
  selectedSentiment = 'All';

  try {
    const [commentsRes, sectionsRes] = await Promise.all([
      fetch(`${API_BASE_URL}/api/comments/${selectedDraftId}`),
      fetch(`${API_BASE_URL}/api/sections/${selectedDraftId}`)
    ]);
    allComments = await commentsRes.json();
    sections = await sectionsRes.json();
  } catch (error) {
    console.error('Error fetching data:', error);
```

```
    }
  }

  function getSentimentColor(score) {
    if (score < 40) return '#dc3545';
    if (score < 60) return '#ffc107';
    return '#28a745';
  }

  function getSentimentText(score) {
    if (score < 40) return 'Negative';
    if (score < 60) return 'Neutral';
    return 'Positive';
  }
</script>

<svelte:head>
  <title>Overall Analysis Dashboard</title>
</svelte:head>

<div class="container">
  <!-- Filters Section -->
  <div class="controls">
    <div class="control-group">
      <label for="draft-select">Draft Selection</label>
      <select id="draft-select" bind:value={selectedDraftId}>
```

```
      <option value="">-- Select a Draft --</option>
      {#each drafts as draft}
        <option value={draft.draft_id}>{draft.title}</option>
      {/each}
    </select>
  </div>

  {#if selectedDraftId}
    <div class="control-group">
      <label for="section-select">Section</label>
      <select id="section-select" bind:value={selectedSectionId}>
        <option value="all">All Sections</option>
        {#each sections as section}
          <option value={section.section_id}>{section.section_title}</option>
        {/each}
      </select>
    </div>

    <div class="control-group">
      <label for="state-select">State</label>
      <select id="state-select" bind:value={selectedState}>
        {#each availableStates as state}
          <option value={state}>{state}</option>
        {/each}
      </select>
    </div>
```

```
      <div class="control-group">
        <label for="action-select">Action Type</label>
        <select id="action-select" bind:value={selectedAction}>
          <option value="All">All</option>
          <option value="Suggest removal">Suggest removal</option>
          <option value="In Agreement">In Agreement</option>
          <option value="Suggest modification">Suggest modification</option>
        </select>
      </div>

      <div class="control-group">
        <label for="sentiment-select">Sentiment</label>
        <select id="sentiment-select" bind:value={selectedSentiment}>
          <option value="All">All</option>
          <option value="Positive">Positive</option>
          <option value="Neutral">Neutral</option>
          <option value="Negative">Negative</option>
        </select>
      </div>
    {/if}
  </div>

  {#if selectedDraftId}
    <!-- Summary Cards -->
    <div class="comment-summary">
```

```
        <div class="comment-box total">
            <h3>Total Comments</h3>
            <p>{totalComments}</p>
        </div>
        <div class="comment-box positive">
            <h3>Positive Comments</h3>
            <p>{positiveCount}</p>
        </div>
        <div class="comment-box neutral">
            <h3>Neutral Comments</h3>
            <p>{neutralCount}</p>
        </div>
        <div class="comment-box negative">
            <h3>Negative Comments</h3>
            <p>{negativeCount}</p>
        </div>
        {#if selectedDraftId}
    <NuanceCards {commentsForDisplay} />
{/if}
        </div>



        <!-- Dashboard Grid -->
        <div class="dashboard-grid">
            <!-- Gauge Chart -->
```

```
        <div class="card gauge-card ">
  <h3>Overall Sentiment Distribution </h3>
  <SentimentChart {commentsForDisplay} />
</div>

        <!-- Action Type Pie Chart -->
        <div class="card action-card">
          <h3>Comments by Action Type</h3>
          <div class="chart-wrapper">
            <canvas use:chart={actionPieChartConfig}></canvas>
          </div>
        </div>

        <!-- Sentiment Bar Chart -->
        <div class="card sentiment-card">
          <h3>Comments by Sentiment Label</h3>
          <div class="chart-wrapper">
            <canvas use:chart={sentimentBarChartConfig}></canvas>
          </div>
        </div>

        <!-- Section Heat Map -->
        <div class="card heatmap-card">
          <h3>Industry × Section Sentiment Heatmap</h3>
          <div class="heatmap-wrapper">
            <Heatmap {commentsForDisplay} />
```

```
  </div>
</div>

<!-- Geographic Map -->
<div class="card map-card">
  <h3>Geographic Distribution</h3>
  <div class="map-container">
    <Map {stateMapData} {selectedDraftId} fetchFromApi={false} />
    <div class="map-legend">
      <div class="legend-item">
        <div class="legend-color" style="background: #28a745;"></div>
        <span>Positive</span>
      </div>
      <div class="legend-item">
        <div class="legend-color" style="background: #007bff;"></div>
        <span>Neutral</span>
      </div>
      <div class="legend-item">
        <div class="legend-color" style="background: #dc3545;"></div>
        <span>Negative</span>
      </div>
    </div>
  </div>
</div>

<!-- Pie Chart -->
```

```
        <div class="card section-card">
          <h3>Comments recevied per Section</h3>
          <div class="chart-wrapper">
            <canvas use:chart={sectionPieChartConfig}></canvas>
          </div>
        </div>

      </div>
      <!-- Draft Summary Section -->
      {#if selectedDraft}
        <div class="summary-section">
          <div class="card summary-card">
            <h3>Draft Summary</h3>
            <div class="summary-content">
              <div class="summary-text">
                <h4>{selectedDraft.title}</h4>
                <p>{selectedDraft.draft_ai_summary || 'No summary available for this
draft.'}</p>
              </div>
              {#if selectedDraft.word_cloud_image_path}
                <div class="wordcloud-container">
                  <img src="{API_BASE_URL}/{selectedDraft.word_cloud_image_path}"
alt="Draft Word Cloud">
                </div>
              {/if}
            </div>
          </div>
```

```
        </div>
      {/if}

      <!-- Section Summary (when specific section selected) -->
      {#if selectedSectionId !== 'all'}
        {@const selectedSection = sections.find(s => s.section_id == selectedSectionId)}
        {#if selectedSection}
        <div class="summary-content">
          <div class="summary-text">
            <h4>{selectedSection.section_title}</h4>
            <p><strong>Summary:</strong> {selectedSection.section_ai_summary || 'No
summary available.'}</p>
            {#if selectedSection.section_ai_key_points}
              <p><strong>Key Points:</strong> {selectedSection.section_ai_key_points}</p>
            {/if}
          </div>
          {#if selectedSection.word_cloud_image_path}
            <div class="wordcloud-container">
              <img src="{API_BASE_URL}/{selectedSection.word_cloud_image_path}"
alt="Section Word Cloud">
            </div>
          {/if}
        </div>
        {/if}
      {/if}

    {:else}
```

```
      <div class="placeholder">
        <svg width="64" height="64" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
          <path d="M14 2H6a2 2 0 0 0-2 2v16a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V8z"/>
          <polyline points="14 2 14 8 20 8"/>
        </svg>
        <h3>No Draft Selected</h3>
        <p>Please select a draft to view the overall analysis</p>
      </div>
    {/if}
</div>

<style>
    :global(body) {
      margin: 0;
      font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, sans-serif;
      background: #f5f7fa;
    }

    .container {
      max-width: 1600px;
      margin: 0 auto;
      padding: 2rem;
    }

    .controls {
```

```
    display: flex;
    gap: 1rem;
    margin-bottom: 2rem;
    flex-wrap: wrap;
    background: white;
    padding: 1.5rem;
    border-radius: 12px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.08);
}

.control-group {
    display: flex;
    flex-direction: column;
    gap: 0.5rem;
    min-width: 200px;
    flex: 1;
}

.control-group label {
    font-size: 0.875rem;
    font-weight: 600;
    color: #495057;
}

select {
    font-size: 0.9rem;
```

```
    padding: 0.625rem 1rem;
    border-radius: 8px;
    border: 2px solid #dee2e6;
    background: white;
    cursor: pointer;
    transition: border-color 0.2s;
}

select:hover {
    border-color: #adb5bd;
}

select:focus {
    outline: none;
    border-color: #0d6efd;
}

.comment-summary {
    display: flex;
    justify-content: space-around;
    /* margin-bottom: 1.5rem; */
    flex-wrap: wrap;
}

.comment-box {
    background: white;
```

```
    border-radius: 10px;
    box-shadow: 0 2px 6px rgba(0, 0, 0, 0.08);
    text-align: center;
    padding:0.5rem 1rem;
    display:flex;
    justify-content:center;
    align-items:center;
    flex-direction:column;
    max-height:85px;

}

.comment-box h3 {
    font-size: 0.9rem;
    font-weight: 600;
    color: #495057;
    margin: 0;
    /* margin-bottom: 0.5rem; */
}

.comment-box p {
    font-size: 1.2rem;
    font-weight: bold;
    color: #343a40;
    margin: 0;
}
```

```
.comment-box.total {
    border-bottom:2px solid #6c757d;
}
.comment-box.total p { color: #6c757d; }

.comment-box.positive {
   border-bottom:2px solid #28a745;
}
.comment-box.positive p { color: #28a745; }

.comment-box.neutral {
    border-bottom:2px solid #007bff;
}
.comment-box.neutral p { color: #007bff; }

.comment-box.negative {
    border-bottom:2px solid #dc3545;
}
.comment-box.negative p { color: #dc3545; }

.summary-section {
   margin-bottom: 2rem;
}
```

```
    .summary-card {
        background: white;
        border-radius: 12px;
        box-shadow: 0 2px 8px rgba(0, 0, 0, 0.08);
        padding: 2rem;

    }

    .summary-card h3 {
        margin: 0 0 1rem 0;
        font-size: 1.25rem;
        color: #2d3748;
        border-bottom: 2px solid #e2e8f0;
        padding-bottom: 0.75rem;
    }
    .summary-content {
    display: flex;
    justify-content: space-between;
    align-items: center;
    gap: 2rem;
}

    .summary-text {
    flex: 1;
    max-width: 700px;
}
```

```
.summary-text h4 {
    margin: 0 0 1rem 0;
    font-size: 1.1rem;
    color: #2d3748;
}

.summary-text p {
    margin: 0 0 1rem 0;
    line-height: 1.7;
    color: #4a5568;
}

    .summary-content p {
        margin: 0 0 1rem 0;
        line-height: 1.7;
        color: #4a5568;
    }

    .wordcloud-container {
    flex-shrink: 0;
    width: 600px;
    display: flex;
    align-items: center;
    justify-content: center;
}
```

```
.wordcloud-container img {
    max-width: 100%;
    max-height: 300px;
    height: auto;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

    .dashboard-grid {
        display: grid;
        grid-template-columns: repeat(3, 1fr);
        gap: 1.5rem;
    }

    .card {
        background: white;
        border-radius: 12px;
        box-shadow: 0 2px 8px rgba(0, 0, 0, 0.08);
        overflow: hidden;
    }

    .card h3 {
        margin: 0;
        padding: 1.25rem;
```

```
    font-size: 1rem;
    font-weight: 600;
    color: #495057;
    border-bottom: 2px solid #f1f3f5;
    background: #fafafa;
    text-align: center;
}

.gauge-card {
    grid-column: span 1;
}

.action-card {
    grid-column: span 1;
}

.sentiment-card {
    grid-column: span 1;
}

.section-card {
    grid-column: span 2;
}

.map-card {
    grid-column: span 1;
```

```
    grid-row: span 2;
}

.heatmap-card {
    grid-column: span 2;
}

.gauge-wrapper {
    padding: 2rem 1rem;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.gauge-canvas-container {
    position: relative;
    width: 100%;
    max-width: 250px;
    height: 125px;
}

.gauge-labels {
    position: relative;
    width: 100%;
    max-width: 250px;
    margin-top: 0.5rem;
```

```
}

.gauge-label {
    position: absolute;
    font-size: 0.75rem;
    color: #6c757d;
    font-weight: 600;
}

.min-label { left: 0; }
.max-label { right: 0; }

.gauge-center {
    text-align: center;

}

.gauge-value {
    font-size: 2rem;
    font-weight: 700;
    margin-bottom: 0.25rem;
}

.gauge-text {
    font-size: 1.1rem;
    font-weight: 600;
```

```
}

.chart-wrapper {
    padding: 1.5rem;
    height: 280px;
}

.map-container {
    padding: 1rem;
    height: 800px;
    display: flex;
    flex-direction: column;

}

.map-legend {
    display: flex;
    justify-content: center;
    gap: 1.5rem;
    margin-top: 1rem;
    padding-top: 1rem;
    border-top: 1px solid #e9ecef;
}

.legend-item {
    display: flex;
```

```
    align-items: center;
    gap: 0.5rem;
}

.legend-color {
    width: 16px;
    height: 16px;
    border-radius: 50%;
    border: 2px solid #fff;
    box-shadow: 0 1px 3px rgba(0, 0, 0, 0.2);
}

.legend-item span {
    font-size: 0.875rem;
    color: #495057;
    font-weight: 500;
}

.heatmap-wrapper {
    padding: 1rem;
    max-height: 600px;
    overflow: auto;
}

.placeholder {
    text-align: center;
```

```
    padding: 5rem 2rem;
    background: white;
    border-radius: 12px;
    box-shadow: 0 2px 8px rgba(0, 0, 0, 0.08);
}

.placeholder svg {
    color: #cbd5e0;
    margin-bottom: 1.5rem;
}

.placeholder h3 {
    margin: 0 0 0.5rem 0;
    color: #2d3748;
    font-size: 1.5rem;
}

.placeholder p {
    margin: 0;
    color: #718096;
    font-size: 1rem;
}

@media (max-width: 1400px) {
    .dashboard-grid {
        grid-template-columns: repeat(2, 1fr);
```

```
    }

    .section-card,
    .heatmap-card {
        grid-column: span 2;
    }

    .map-card {
        grid-column: span 2;
        grid-row: span 1;
    }
}

@media (max-width: 768px) {
    .container {
        padding: 1rem;
    }

    .dashboard-grid {
        grid-template-columns: 1fr;
    }

    .gauge-card,
    .action-card,
    .sentiment-card,
    .section-card,
```

```
      .map-card,
      .heatmap-card {
        grid-column: span 1;
        grid-row: span 1;
      }
  }
</style>
```

“ and this is filterStore.js “// src/lib/stores/filterStore.js

```
import { writable, derived } from 'svelte/store';

const HIERARCHY = ['industry', 'section', 'state', 'action', 'sentiment'];

const createFilterStore = () => {
  const { subscribe, set, update } = writable({
    state: null,
    section: null,    // will store section_id (stable)
    action: null,
    sentiment: null,
    industry: null,
    drillDownPath: [] // e.g., ['industry','section']
  });

  function clearChildren(filtersObj, parentFilterType) {
    const idx = HIERARCHY.indexOf(parentFilterType);
    if (idx === -1) return filtersObj;
    const newFilters = { ...filtersObj };
```

```
    HIERARCHY.slice(idx + 1).forEach(ft => {
      newFilters[ft] = null;
    });
    // Trim drill path to parent index
    newFilters.drillDownPath = (newFilters.drillDownPath || []).filter(f => HIERARCHY.indexOf(f) <= idx);
    return newFilters;
  }

  return {
    subscribe,
    setFilter: (filterType, value) => {
      // backward-compatible, simple set
      update(filters => {
        if (filters[filterType] === value) {
          const newPath = filters.drillDownPath.filter(f => f !== filterType);
          return { ...filters, [filterType]: null, drillDownPath: newPath };
        }
        const newPath = filters.drillDownPath.includes(filterType) ? filters.drillDownPath : [...filters.drillDownPath, filterType];
        // clear children of this filter when user sets it (prevents stale deeper filters)
        let next = { ...filters, [filterType]: value, drillDownPath: newPath };
        next = clearChildren(next, filterType);
        return next;
      });
    },
```

```
// More explicit drill-down / hierarchical set
setDrillDownFilter: (filterType, value, parentFilterType = null) => {
  update(filters => {

    // Toggle: same value clicked -> clear
    if (filters[filterType] === value) {
      const newPath = filters.drillDownPath.filter(f => f !== filterType);
      // Also clear children of this filter (if any)
      let cleared = { ...filters, [filterType]: null, drillDownPath: newPath };
      cleared = clearChildren(cleared, filterType);
      return cleared;
    }

    // Build new state: clear any children deeper than this filter
    let next = { ...filters, [filterType]: value };
    next = clearChildren(next, filterType);

    // Build drill path: keep everything up to this filter, then add it
    const desiredIndex = HIERARCHY.indexOf(filterType);
    let newPath = (filters.drillDownPath || []).filter(f => HIERARCHY.indexOf(f) < desiredIndex);
    if (!newPath.includes(filterType)) newPath.push(filterType);
    next.drillDownPath = newPath;

    // If parentFilterType provided, ensure parent exists (no-op if not)
    if (parentFilterType) {
      const parentIdx = HIERARCHY.indexOf(parentFilterType);
```

```
      if (parentIdx !== -1) {
        // keep only up to parent in path, then add this filter
        newPath = newPath.filter(f => HIERARCHY.indexOf(f) <= parentIdx);
        if (!newPath.includes(parentFilterType)) newPath.push(parentFilterType);
        if (!newPath.includes(filterType)) newPath.push(filterType);
        next.drillDownPath = newPath;
      }
    }

    return next;
  });
},

clearFilter: (filterType) => {
  update(filters => {
    // Clear given filter and children of that filter
    let next = { ...filters, [filterType]: null };
    next = clearChildren(next, filterType);
    next.drillDownPath = (next.drillDownPath || []).filter(f => f !== filterType);
    return next;
  });
},

clearAllFilters: () => {
  set({
    state: null,
```

```
    section: null,
    action: null,
    sentiment: null,
    industry: null,
    drillDownPath: []
  });
},

reset: () => {
  set({
    state: null,
    section: null,
    action: null,
    sentiment: null,
    industry: null,
    drillDownPath: []
  });
},

// optional convenience to move one level up
drillUp: () => {
  update(filters => {
    const path = (filters.drillDownPath || []);
    if (!path.length) return filters;
    const last = path[path.length - 1];
    const newFilters = { ...filters, [last]: null, drillDownPath: path.slice(0, -1) };
```

```
      // also clear children of new last if any
      if (newFilters.drillDownPath.length) {
        const newLast = newFilters.drillDownPath[newFilters.drillDownPath.length - 1];
        return clearChildren(newFilters, newLast);
      }
      return newFilters;
    });
  }
 };
};

export const filterStore = createFilterStore();

export const activeFilterCount = derived(
 filterStore,
 $filters => Object.entries($filters)
  .filter(([key, value]) => key !== 'drillDownPath' && value !== null)
  .length
);

“ and this is chartAction.js “   // Frontend/src/lib/chartAction.js
  import Chart from 'chart.js/auto';
  import ChartDataLabels from 'chartjs-plugin-datalabels';
  import { MatrixController, MatrixElement } from 'chartjs-chart-matrix';

  // Register plugins and controllers
```

```
  Chart.register(ChartDataLabels);
  Chart.register(MatrixController, MatrixElement);

  /**
   * A Svelte Action for integrating Chart.js in a robust, lifecycle-aware way.
   * @param {HTMLCanvasElement} canvas The canvas element the action is applied to.
   * @param {import('chart.js').ChartConfiguration} config The Chart.js configuration object.
   */
  export function chart(canvas, config) {
    let chartInstance = new Chart(canvas, config);

    return {
      // This function is called whenever the 'config' parameter changes.
      update(newConfig) {
        // Destroy the existing chart instance to prevent memory leaks and ensure proper
updates
        chartInstance.destroy();
        // Create a new chart instance with the updated configuration
        chartInstance = new Chart(canvas, newConfig);
      },

      // This function is called when the component is unmounted, preventing memory leaks.
      destroy() {
        chartInstance.destroy();
      }
    };
```

```
  }
```

“ and this is heatmap.svelte “<script>

```
  import { createEventDispatcher } from 'svelte';

  const dispatch = createEventDispatcher();

  export let commentsForDisplay = [];

  let heatmapData = [];
  let industries = [];
  let sections = [];
  let maxSectionLength = 30;

  $: {
    const dataMap = {};

    commentsForDisplay.forEach(c => {
      if (!c.section_title) return;

      const industry = c.industry || 'Individual';
      const section = c.section_title.length > maxSectionLength
        ? c.section_title.substring(0, maxSectionLength) + '...'
        : c.section_title;

      if (!dataMap[industry]) {
        dataMap[industry] = {};
```

```
    }
    if (!dataMap[industry][section]) {
        dataMap[industry][section] = { sum: 0, count: 0 };
    }

    let score = 0;
    if (c.sentiment_label === 'Positive') score = 1;
    if (c.sentiment_label === 'Neutral') score = 0;
    if (c.sentiment_label === 'Negative') score = -1;

    dataMap[industry][section].sum += score;
    dataMap[industry][section].count++;
});

industries = Object.keys(dataMap).sort((a, b) => {
    if (a === 'Individual') return -1;
    if (b === 'Individual') return 1;
    return a.localeCompare(b);
});

sections = [...new Set(
    industries.flatMap(ind => Object.keys(dataMap[ind]))
)].sort();

heatmapData = industries.map(ind =>
    sections.map(sec => {
```

```
            const data = dataMap[ind]?.[sec];
            if (data) {
                return data.sum / data.count;
            }
            return null;
        })
    );
}

function getColor(value) {
    if (value === null) return '#e9ecef';
    if (value > 0.5) return '#28a745';
    if (value > 0.1) return '#90ee90';
    if (value >= -0.1) return '#007bff';
    if (value > -0.5) return '#ffb3b3';
    return '#dc3545';
}

function getSentimentText(value) {
    if (value === null) return 'No data';
    if (value > 0.3) return 'Positive';
    if (value < -0.3) return 'Negative';
    return 'Neutral';
}

function handleCellClick(industry, section) {
```

```
    dispatch('cellClick', { industry, section });
  }
</script>

<div class="heatmap-container">
  {#if industries.length === 0}
    <p class="no-data">No data available for heatmap</p>
  {:else}
    <div class="heatmap-wrapper">
      <div class="heatmap-scroll">
        <table class="heatmap-table">
          <thead>
            <tr>
              <th class="corner-cell">Industry / Section</th>
              {#each sections as section}
                <th class="section-header" title={section}>
                  <div class="header-text">{section}</div>
                </th>
              {/each}
            </tr>
          </thead>
          <tbody>
            {#each industries as industry, i}
              <tr>
                <th class="industry-header">{industry}</th>
                {#each heatmapData[i] as value, j}
```

```
                        <td
                          class="heatmap-cell"
                          class:clickable={value !== null}
                          style:background-color={getColor(value)}
                          title="{industry} ×
{sections[j]}&#10;{getSentimentText(value)}&#10;Score: {value !== null ? value.toFixed(2) :
'N/A'}&#10;{value !== null ? 'Click to filter' : ''}"
                          on:click={() => value !== null && handleCellClick(industry, sections[j])}
                        >
                          {#if value !== null}
                            <span class="cell-value">{value.toFixed(2)}</span>
                          {:else}
                            <span class="cell-empty">—</span>
                          {/if}
                        </td>
                      {/each}
                    </tr>
                  {/each}
                </tbody>
              </table>
            </div>
          </div>

          <div class="legend">
            <span class="legend-title">Sentiment Scale:</span>
            <div class="legend-item">
              <div class="legend-color" style="background: #28a745;"></div>
```

```
            <span>Strong Positive (&gt;0.5)</span>
          </div>
          <div class="legend-item">
            <div class="legend-color" style="background: #90ee90;"></div>
            <span>Positive (0.1 to 0.5)</span>
          </div>
          <div class="legend-item">
            <div class="legend-color" style="background: #007bff;"></div>
            <span>Neutral (-0.1 to 0.1)</span>
          </div>
          <div class="legend-item">
            <div class="legend-color" style="background: #ffb3b3;"></div>
            <span>Negative (-0.5 to -0.1)</span>
          </div>
          <div class="legend-item">
            <div class="legend-color" style="background: #dc3545;"></div>
            <span>Strong Negative (&lt;-0.5)</span>
          </div>
          <div class="legend-note">
            <em>💡 Click on any cell to filter by industry and section</em>
          </div>
        </div>
    {/if}
</div>

<style>
```

```
.heatmap-container {
    height: 100%;
    display: flex;
    flex-direction: column;
    padding: 1rem;
    width: 100%;
}

.heatmap-wrapper {
    flex: 1;
    overflow: auto;
    border: 1px solid #dee2e6;
    border-radius: 8px;
    background: white;
    width: 100%;
}

.heatmap-scroll {
    overflow: auto;
    max-height: 100%;
    width:100%;
}

.heatmap-table {
    border-collapse: separate;
    border-spacing: 0;
```

```
    width: 100%;
    font-size: 0.85rem;
}

.corner-cell {
    background: #f8f9fa;
    padding: 0.75rem;
    text-align: left;
    font-weight: 600;
    color: #495057;
    border-bottom: 2px solid #dee2e6;
    border-right: 2px solid #dee2e6;
    position: sticky;
    left: 0;
    top: 0;
    z-index: 3;
    min-width: 120px;
}

.section-header {
    background: #f8f9fa;
    padding: 0.5rem 0.5rem;
    text-align: center;
    font-weight: 600;
    color: #495057;
    border-bottom: 2px solid #dee2e6;
```

```
    border-right: 1px solid #e9ecef;
    position: sticky;
    top: 0;
    z-index: 2;
    min-width: 60px;
    width: fit-content;
    height: auto;
}

.header-text {
    writing-mode: horizontal;
    font-size: 0.75rem;
}

.industry-header {
    background: #f8f9fa;
    padding: 0.75rem;
    text-align: left;
    font-weight: 600;
    color: #495057;
    white-space: nowrap;
    position: sticky;
    left: 0;
    z-index: 1;
    border-right: 2px solid #dee2e6;
    border-bottom: 1px solid #e9ecef;
```

```
    min-width: 120px;
}

.heatmap-cell {
    min-width: 60px;
    max-width: 60px;
    height: 50px;
    text-align: center;
    vertical-align: middle;
    border-right: 1px solid #e9ecef;
    border-bottom: 1px solid #e9ecef;
    transition: all 0.2s;
}

.heatmap-cell.clickable {
    cursor: pointer;
}

.heatmap-cell.clickable:hover {
    box-shadow: inset 0 0 0 3px #0d6efd;
    z-index: 1;
    transform: scale(1.05);
}

.cell-value {
    font-size: 0.75rem;
```

```
    font-weight: 600;
    color: #000;
    text-shadow: 0 0 3px rgba(255,255,255,0.8);
}

.cell-empty {
    color: #adb5bd;
    font-size: 1rem;
}

.legend {
    display: flex;
    justify-content: center;
    align-items: center;
    gap: 1rem;
    padding: 1rem;
    margin-top: 1rem;
    border-top: 2px solid #e9ecef;
    flex-wrap: wrap;
    background: #f8f9fa;
    border-radius: 6px;
}

.legend-title {
    font-weight: 600;
    color: #495057;
```

```
    margin-right: 0.5rem;
}

.legend-item {
    display: flex;
    align-items: center;
    gap: 0.4rem;
    font-size: 0.75rem;
    color: #495057;
}

.legend-color {
    width: 20px;
    height: 20px;
    border-radius: 4px;
    border: 1px solid #dee2e6;
}

.legend-note {
    width: 100%;
    text-align: center;
    color: #0d6efd;
    font-size: 0.8rem;
    margin-top: 0.5rem;
}
```

```
    .no-data {
        text-align: center;
        padding: 3rem;
        color: #6c757d;
        font-style: italic;
    }

    .heatmap-scroll::-webkit-scrollbar {
        width: 10px;
        height: 10px;
    }

    .heatmap-scroll::-webkit-scrollbar-track {
        background: #f1f3f5;
        border-radius: 5px;
    }

    .heatmap-scroll::-webkit-scrollbar-thumb {
        background: #ced4da;
        border-radius: 5px;
    }

    .heatmap-scroll::-webkit-scrollbar-thumb:hover {
        background: #adb5bd;
    }
</style>
```

“ and this is impactful comments “<script>

```
  export let commentsForDisplay = [];

  // The minimum number of comments required to show the "impactful" list.
  const MIN_COMMENTS_THRESHOLD = 3;

  $: topComments = (() => {
    // 1. Filter out comments that don't have an AI summary, as they can't be displayed.
    const relevantComments = commentsForDisplay.filter(c => c.ai_summary);

    // 2. Separate comments into 'Positive' and 'Negative' based on their sentiment_label.
    const positiveComments = relevantComments.filter(c => c.sentiment_label === 'Positive');
    const negativeComments = relevantComments.filter(c => c.sentiment_label === 'Negative');

    let topPositive = [];
    let topNegative = [];

    // 3. Process POSITIVE comments
    // Check if we have enough positive comments to meet the threshold.
    if (positiveComments.length >= MIN_COMMENTS_THRESHOLD) {
      // Sort by the 'score_positive' in descending order (highest score first).
      const sortedPositive = positiveComments.sort((a, b) =>
        (b.score_positive || 0) - (a.score_positive || 0)
      );
      // Take the top 3.
      topPositive = sortedPositive.slice(0, 3);
```

```
    }

    // 4. Process NEGATIVE comments
    // Check if we have enough negative comments to meet the threshold.
    if (negativeComments.length >= MIN_COMMENTS_THRESHOLD) {
      // Sort by the 'score_negative' in descending order (highest score first).
      const sortedNegative = negativeComments.sort((a, b) =>
        (b.score_negative || 0) - (a.score_negative || 0)
      );
      // Take the top 3.
      topNegative = sortedNegative.slice(0, 3);
    }

    // 5. Return the final lists. They will be empty if the threshold isn't met.
    return { topPositive, topNegative };
  })();
</script>

<div class="impactful-comments">
  <div class="comment-section positive-section">
    <h4>Most Positive Comments</h4>

    {#if topComments.topPositive.length > 0}
      {#each topComments.topPositive as comment}
        <div class="comment-card positive">
          <div class="comment-header">
```

```
            <span class="section-badge">{comment.section_title}</span>
            <span class="action-badge">{comment.action_type}</span>
            <span class="sentiment-score">
              👍 {comment.score_positive?.toFixed(2)}
            </span>
          </div>
          <p class="comment-summary">{comment.ai_summary}</p>
          <div class="comment-meta">
            <span class="industry">{comment.industry || 'Individual'}</span>
            <span class="state">{comment.state}</span>
          </div>
        </div>
      {/each}
    {:else}
      <div class="no-comments-message">
        Not enough positive comments in this selection to display a top list.
      </div>
    {/if}
  </div>

  <div class="comment-section negative-section">
    <h4>Most Critical Comments</h4>

    {#if topComments.topNegative.length > 0}
      {#each topComments.topNegative as comment}
        <div class="comment-card negative">
```

```
            <div class="comment-header">
              <span class="section-badge">{comment.section_title}</span>
              <span class="action-badge">{comment.action_type}</span>
              <span class="sentiment-score">
                👎 {comment.score_negative?.toFixed(2)}
              </span>
            </div>
            <p class="comment-summary">{comment.ai_summary}</p>
            <div class="comment-meta">
              <span class="industry">{comment.industry || 'Individual'}</span>
              <span class="state">{comment.state}</span>
            </div>
          </div>
        {/each}
      {:else}
        <div class="no-comments-message">
          Not enough negative comments in this selection to display a top list.
        </div>
      {/if}
    </div>
</div>
<style>
    .impactful-comments {
      display: grid;
      grid-template-columns: 1fr 1fr;
      gap: 1.5rem;
```

```
    }

    .comment-section h4 {
        margin: 0 0 1rem 0;
        padding: 0.75rem;
        background: #fafafa;
        border-radius: 8px;
        text-align: center;
        font-size: 0.95rem;
        font-weight: 600;
        color: #495057;
    }
    /* ... add this at the end of your existing styles ... */

.no-comments-message {
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
    padding: 2rem;
    height: 100%;
    color: #6c757d;
    background-color: #f8f9fa;
    border-radius: 8px;
    font-style: italic;
    font-size: 0.9rem;
```

```
}

    .comment-card {
        background: white;
        border-radius: 8px;
        padding: 1rem;
        margin-bottom: 1rem;
        box-shadow: 0 2px 4px rgba(0, 0, 0, 0.06);
        border-left: 4px solid;
    }

    .comment-card.positive {
        border-left-color: #28a745;
        background-color: #DBFAE6;
    }

    .comment-card.negative {
        border-left-color: #dc3545;
        background-color: #FDE8E8;
    }

    .comment-header {
        display: flex;
        justify-content: space-between;
        align-items: center; /* Vertically align items in the header */
        margin-bottom: 0.75rem;
```

```
    gap: 0.5rem;
}

.section-badge, .action-badge {
    font-size: 0.7rem;
    padding: 0.25rem 0.5rem;
    border-radius: 4px;
    font-weight: 600;
}

.section-badge {
    /* background: #e3f2fd; */
    color: #1976d2;
    flex: 1;
    overflow: hidden;
    text-overflow: ellipsis;
    white-space: nowrap;
}

.action-badge {
    background: #f5f5f5;
    color: #666;
}

.comment-summary {
    font-size: 0.85rem;
```

```
    line-height: 1.5;
    color: #333;
    margin: 0 0 0.75rem 0;
}

.comment-meta {
    display: flex;
    justify-content: space-between;
    font-size: 0.75rem;
    color: #6c757d;
}

.industry {
    font-weight: 600;
}

.sentiment-score {
    display: flex;
    align-items: center;
    gap: 0.25rem;
    font-size: 0.75rem;
    font-weight: 600;
    border-radius: 4px;
    padding: 0.25rem 0.5rem;
}
```

```
    .comment-card.positive .sentiment-score {
        background-color: rgba(40, 167, 69, 0.1);
        color: #28a745;
    }

    .comment-card.negative .sentiment-score {
        background-color: rgba(220, 53, 69, 0.1); /* Light red */
        color: #dc3545;
    }

    .sentiment-score .icon {
        font-size: 0.8rem;
    }
</style>
```

“ and this is map.svelte “

```
<script>
    import { onMount, onDestroy, createEventDispatcher } from 'svelte';

    const dispatch = createEventDispatcher();

    export let stateMapData = null;
    export let selectedDraftId = null;
    export let fetchFromApi = false;

    let map;
    let layerGroup;
    let L;
```

```
const stateCentroids = {
    'Karnataka': [12.9716, 77.5946],
    'Delhi': [28.7041, 77.1025],
    'Maharashtra': [19.7515, 75.7139],
    'Gujarat': [22.2587, 71.1924],
    'Telangana': [17.3850, 78.4867],
    'Tamil Nadu': [13.0827, 80.2707],
    'Haryana': [29.0588, 76.0856],
    'West Bengal': [22.9868, 87.8550],
    'Uttar Pradesh': [26.8467, 80.9462],
    'Rajasthan': [27.0238, 74.2179],
    'Madhya Pradesh': [22.9734, 78.6569],
    'Kerala': [10.8505, 76.2711],
    'Andhra Pradesh': [15.9129, 79.73999]
};

function getColor(d) {
    if (!d || !d.total) return '#AAAAAA';
    const score = ((d.positive || 0) - (d.negative || 0)) / d.total * 100;
    if (score < -5) return '#FF4136';
    if (score > 5) return '#3D9970';
    return '#00A2E8';
}

function radiusFromTotal(total) {
```

```
    return 6 + Math.sqrt(total || 0) * 3;
}

async function initMap() {
    const leafletModule = await import('leaflet');
    L = leafletModule.default || leafletModule;
    await import('leaflet/dist/leaflet.css');

    map = L.map('leaflet-map', {
        center: [22.0, 80.0],
        zoom: 5,
        attributionControl: false
    });

    L.tileLayer('https://tile.openstreetmap.org/{z}/{x}/{y}.png', {
        subdomains: 'abcd',
        maxZoom: 19
    }).addTo(map);

    layerGroup = L.layerGroup().addTo(map);

    if (fetchFromApi && selectedDraftId) {
        await fetchData();
    } else {
        renderFromProp();
    }
```

```
}

async function fetchData() {
    if (!selectedDraftId) return;
    try {
        const resp = await fetch(`/api/map-data/${selectedDraftId}`);
        if (!resp.ok) return;
        const arr = await resp.json();
        const obj = {};
        arr.forEach(it => {
            obj[it.state] = {
                total: it.total,
                positive: it.positive,
                neutral: it.neutral,
                negative: it.negative,
                lat: it.lat,
                lon: it.lon
            };
        });
        stateMapData = obj;
        renderFromProp();
    } catch (err) {
        console.error('Error fetching map-data:', err);
    }
}
```

```
function renderFromProp() {
    if (!layerGroup || !stateMapData) return;
    layerGroup.clearLayers();

    Object.entries(stateMapData).forEach(([state, d]) => {
        let lat = d.lat, lon = d.lon;
        if ((lat === undefined || lon === undefined) && stateCentroids[state]) {
            [lat, lon] = stateCentroids[state];
        }
        if (!lat || !lon) return;

        const circle = L.circleMarker([lat, lon], {
            radius: radiusFromTotal(d.total),
            fillColor: getColor(d),
            color: '#FFFFFF',
            weight: 2,
            fillOpacity: 0.85
        }).addTo(layerGroup);

        circle.bindPopup(`
            <strong>${state}</strong><br/>
            Total: ${d.total || 0}<br/>
            Positive: ${d.positive || 0}<br/>
            Neutral: ${d.neutral || 0}<br/>
            Negative: ${d.negative || 0}<br/>
            <em style="color: #0d6efd; font-size: 0.85em;">Click marker to filter by state</em>
```

```
        `);

        // Add click event for cross-filtering
        circle.on('click', () => {
            dispatch('stateClick', state);
        });

        // Add hover effect
        circle.on('mouseover', function() {
            this.setStyle({
                weight: 3,
                fillOpacity: 1
            });
        });

        circle.on('mouseout', function() {
            this.setStyle({
                weight: 2,
                fillOpacity: 0.85
            });
        });
    });
}

$: if (!fetchFromApi && stateMapData && layerGroup) {
    renderFromProp();
```

```
    }

    $: if (fetchFromApi && selectedDraftId && layerGroup) {
        fetchData();
    }

    onMount(() => {
        initMap();
    });

    onDestroy(() => {
        if (map) map.remove();
    });
</script>

<style>
    #leaflet-map {
        width: 100%;
        height: 100%;
        border-radius: 8px;
        cursor: pointer;
    }
</style>

<div id="leaflet-map"></div>
```

And this is nuancecard.svelte “<script>

```
    export let commentsForDisplay = [];

    $: sarcasticAgreements = commentsForDisplay.filter(
        c => c.action_type === 'In Agreement' && c.sentiment_label === 'Negative'
    ).length;

    $: positiveModifications = commentsForDisplay.filter(
        c => c.action_type === 'Suggest modification' && c.sentiment_label === 'Positive'
    ).length;

    $: negativeRemovals = commentsForDisplay.filter(
        c => c.action_type === 'Suggest removal' && c.sentiment_label === 'Negative'
    ).length;
</script>

<div class="nuance-cards">
    <div class="nuance-card sarcastic">
        <h4>Sarcastic Agreements Detected</h4>
        <div class="card-value">{sarcasticAgreements}</div>
        <p class="card-description">AI detected hidden negativity</p>
    </div>

    <div class="nuance-card positive-mod">
        <h4>Positive Modifications</h4>
        <div class="card-value">{positiveModifications}</div>
        <p class="card-description">Constructive suggestions</p>
```

```
    </div>

    <div class="nuance-card negative-removal">
      <h4>Strongly Opposed</h4>
      <div class="card-value">{negativeRemovals}</div>
      <p class="card-description">Critical removal requests</p>
    </div>
  </div>

  <style>
    .nuance-cards {
      display: grid;
      grid-template-columns: repeat(3, 1fr);
      gap: 1rem;
      margin-bottom: 1.5rem;
    }

    .nuance-card {
      background: white;
      border-radius: 10px;
      box-shadow: 0 2px 6px rgba(0, 0, 0, 0.08);
      padding: 0.5rem;
      text-align: center;
      transition: transform 0.2s;
    }
```

```
.nuance-card:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.12);
}

.card-icon {
    font-size: 2rem;
    margin-bottom: 0.5rem;
}

.nuance-card h4 {
    font-size: 0.85rem;
    font-weight: 600;
    color: #495057;
    margin: 0.5rem 0;
}

.card-value {
    font-size: 1.25rem;
    font-weight: 700;
    margin: 0.5rem 0;
}

.card-description {
    font-size: 0.75rem;
    color: #6c757d;
```

```
    margin: 0;
}

.sarcastic {
    border-bottom: 3px solid #ff9800;
}

.sarcastic .card-value {
    color: #ff9800;
}

.positive-mod {
    border-bottom: 3px solid #28a745;
}

.positive-mod .card-value {
    color: #28a745;
}

.negative-removal {
    border-bottom: 3px solid #dc3545;
}

.negative-removal .card-value {
    color: #dc3545;
}
```

</style>

“ and this is backend app.py “import sqlite3

from flask import Flask, jsonify, send_from_directory

from flask_cors import CORS

import os

app = Flask(__name__, static_folder=os.path.join(os.path.dirname(os.path.abspath(__file__)), 'static'))

CORS(app) # Allow all origins for simplicity in the hackathon

DATABASE_FILE = 'econsultation.db'

def get_db_connection():

    """Creates a database connection with dictionary-like row access."""

    conn = sqlite3.connect(DATABASE_FILE)

    conn.row_factory = sqlite3.Row

    return conn

--- API Endpoints ---

@app.route('/api/drafts', methods=['GET'])

def get_drafts():

    """

    UPGRADED: Returns a list of all drafts, now including all AI analysis columns.

    """

    conn = get_db_connection()

    # Using SELECT * is the easiest way to include all the new columns

```
    drafts = conn.execute('SELECT * FROM drafts').fetchall()
    conn.close()
    return jsonify([dict(row) for row in drafts])

@app.route('/api/sections/<int:draft_id>')
def get_sections_for_draft(draft_id):
    """
    UPGRADED: Returns all sections for a specific draft, including all new AI analysis columns.
    """
    conn = get_db_connection()
    # Using SELECT * to get all columns, including new ones
    sections = conn.execute(
        'SELECT * FROM sections WHERE draft_id = ?',
        (draft_id,)
    ).fetchall()
    conn.close()
    return jsonify([dict(row) for row in sections])

@app.route('/api/comments/<int:draft_id>')
def get_comments_for_draft(draft_id):
    """
    UPGRADED: Returns all comments for a specific draft, joined with user and section data.
    This is the primary endpoint for the interactive dashboard.
    """
    conn = get_db_connection()
    comments = conn.execute("""
```

```
        SELECT
            c.*, -- Selects all comment data, including new score columns
            sec.section_title,
            u.state,
            CASE
                WHEN u.industry IS NULL OR u.industry = '' THEN 'Individual'
                ELSE u.industry
            END as industry,
            u.is_organization
        FROM comments c
        JOIN submissions s ON c.submission_id = s.submission_id
        JOIN sections sec ON c.section_id = sec.section_id
        JOIN users u ON s.user_id = u.user_id
        WHERE s.draft_id = ?
    """, (draft_id,)).fetchall()
    conn.close()
    return jsonify([dict(row) for row in comments])

# --- RESTORED & MAINTAINED from your original code ---

@app.route('/api/drafts/<int:draft_id>', methods=['GET'])
def get_draft_details(draft_id):
    """
    RESTORED: Provides a deeply nested JSON object for a single draft,
    including all its sections and their respective comments.
    """
```

```
    conn = get_db_connection()
    draft = conn.execute('SELECT * FROM drafts WHERE draft_id = ?', (draft_id,)).fetchone()
    if draft is None: return jsonify({"error": "Draft not found"}), 404

    sections = conn.execute('SELECT * FROM sections WHERE draft_id = ? ORDER BY section_id',
(draft_id,)).fetchall()
    sections_list = []
    for section in sections:
        section_dict = dict(section)
        # Your original query had a slight bug in the JOIN condition, I've corrected it.
        # It should join on c.submission_id = s.submission_id, not section_id.
        comments = conn.execute('''
            SELECT c.*, u.first_name, u.last_name, u.organization_name FROM comments c
            JOIN submissions s ON c.submission_id = s.submission_id
            JOIN users u ON s.user_id = u.user_id WHERE c.section_id = ?
        ''', (section['section_id'],)).fetchall()
        section_dict['comments'] = [dict(comment) for comment in comments]
        sections_list.append(section_dict)

    conn.close()
    result = dict(draft)
    result['sections'] = sections_list
    return jsonify(result)

@app.route('/wordclouds/<path:subfolder>/<path:filename>')
def serve_wordcloud(subfolder, filename):
```

```
    """
    RESTORED: A dedicated route for serving word cloud images, which may be
    cleaner for some frontend paths than using the main /static route.
    """
    # Note: Flask's send_from_directory is relative to the *app root*, not the static folder path.
    # We construct a path to the top-level 'static' directory.
    static_dir = os.path.join(os.path.dirname(os.path.abspath(__file__)), '..','static')

    return send_from_directory(static_dir, f'wordclouds/{subfolder}/{filename}')

# Added the generic /static route as well, as it is also very useful.
@app.route('/static/<path:path>')
def serve_static(path):
    return send_from_directory(app.static_folder, path)

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

and this is database setup.py file “import sqlite3

```
import os

DATABASE_FILE = "econsultation.db"

# --- Schema Definition based on your final PDF ---
# Using multiline strings to hold the CREATE TABLE statements
CREATE_DRAFTS_TABLE = """
CREATE TABLE IF NOT EXISTS drafts (
```

```
    draft_id INTEGER PRIMARY KEY,
    title TEXT NOT NULL,
    description TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    draft_ai_summary TEXT,
    summary_positive TEXT,
    summary_negative TEXT,
    summary_neutral TEXT,
    word_cloud_image_path TEXT,
    wordcloud_positive_path TEXT,
    wordcloud_negative_path TEXT,
    wordcloud_neutral_path TEXT
);
"""

CREATE_SECTIONS_TABLE = """
CREATE TABLE IF NOT EXISTS sections (
    section_id INTEGER PRIMARY KEY,
    draft_id INTEGER NOT NULL,
    section_title TEXT NOT NULL UNIQUE,
    section_content TEXT,
    section_ai_summary TEXT,
    section_ai_key_points TEXT,
    summary_positive TEXT,
    summary_negative TEXT,
    summary_neutral TEXT,
```

```
    word_cloud_image_path TEXT,
    wordcloud_positive_path TEXT,
    wordcloud_negative_path TEXT,
    wordcloud_neutral_path TEXT,
    FOREIGN KEY (draft_id) REFERENCES drafts(draft_id)
);
"""

CREATE_USERS_TABLE = """
CREATE TABLE IF NOT EXISTS users (
    user_id INTEGER PRIMARY KEY,
    first_name TEXT NOT NULL,
    last_name TEXT NOT NULL,
    email TEXT NOT NULL UNIQUE,
    phone TEXT,
    address TEXT,
    country TEXT,
    state TEXT,
    is_organization BOOLEAN DEFAULT 0,
    organization_name TEXT,
    industry TEXT CHECK(industry IN ('Education', 'Healthcare', 'Finance', 'Technology',
'Government', 'Non-Profit', 'Other')),
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP
);
"""
```

```
CREATE_SUBMISSIONS_TABLE = """
CREATE TABLE IF NOT EXISTS submissions (
    submission_id INTEGER PRIMARY KEY,
    user_id INTEGER NOT NULL,
    draft_id INTEGER NOT NULL,
    otp_verified BOOLEAN DEFAULT 0,
    submission_status TEXT NOT NULL DEFAULT 'completed',
    submitted_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id),
    FOREIGN KEY (draft_id) REFERENCES drafts(draft_id)
);
"""

CREATE_COMMENTS_TABLE = """
CREATE TABLE IF NOT EXISTS comments (
    comment_id INTEGER PRIMARY KEY,
    submission_id INTEGER NOT NULL,
    section_id INTEGER NOT NULL,
    action_type TEXT NOT NULL CHECK(action_type IN ('In Agreement', 'Suggest removal',
'Suggest modification', 'Implicit Agreement')),
    comment_text TEXT,
    created_at DATETIME DEFAULT CURRENT_TIMESTAMP,
    updated_at DATETIME,
    sentiment_label TEXT,
    sentiment_score REAL,
    score_positive REAL,
```

```
    score_negative REAL,
    score_neutral REAL,
    ai_summary TEXT,
    word_cloud_image_path TEXT,
    FOREIGN KEY (submission_id) REFERENCES submissions(submission_id),
    FOREIGN KEY (section_id) REFERENCES sections(section_id)
);
"""

# --- Data Insertion Queries ---
# All the INSERT statements you need to populate the database with fake data.
INSERT_DATA = """
-- The SQL INSERT statements for drafts, sections, users, submissions, and comments go here.
-- This is a very long string, so it's kept separate for clarity.
INSERT INTO drafts (draft_id, title, description) VALUES
(1, 'The Digital India Act, 2025', 'A comprehensive framework for data protection, intermediary liability, and regulation of emerging technologies.'),
(2, 'The Corporate Governance and ESG Disclosure Bill, 2025', 'A bill to enforce stricter corporate governance norms and mandatory Environmental, Social, and Governance (ESG) reporting.'),
(3, 'The National Green Energy Framework, 2025', 'Policy outlining subsidies, carbon taxes, and infrastructure development for renewable energy.');

INSERT INTO sections (section_id, draft_id, section_title, section_content) VALUES
(1, 1, 'Section 5: Data Localisation Mandates', 'All sensitive personal data of Indian citizens must be stored exclusively on servers located within India.'),
(2, 1, 'Section 12: Intermediary Liability and Safe Harbour', 'Redefines the conditions under which social media platforms are granted safe harbour protections.'),
```

(3, 1, 'Section 22: Regulation of AI Models', 'Establishes a regulatory body to classify and audit high-risk AI systems deployed in critical sectors.'),

(4, 2, 'Section 8: Mandatory ESG Reporting', 'All listed companies with a turnover above Rs. 500 Cr must publish an annual ESG report based on a standardized format.'),

(5, 2, 'Section 15: Independent Director Quotas', 'Mandates that at least 50% of the board members of listed companies must be Independent Directors.'),

(6, 2, 'Section 21: Executive Compensation Caps', 'Introduces a cap on the cash component of executive compensation, linking it to the median employee salary.'),

(7, 3, 'Section 4: Solar Power Subsidies', 'Details the subsidy structure for rooftop solar panel installations for residential and commercial properties.'),

(8, 3, 'Section 9: Carbon Tax Implementation', 'A phased implementation of a carbon tax on industries based on their greenhouse gas emissions.'),

(9, 3, 'Section 14: Electric Vehicle Charging Infrastructure', 'Mandates all new commercial buildings to allocate 10% of parking space for EV charging stations.');

INSERT INTO users (user_id, first_name, last_name, email, phone, address, country, state, is_organization, organization_name, industry) VALUES

(1, 'Priya', 'Sharma', 'priya.sharma@email.com', '9876543210', 'Indiranagar, Bangalore', 'India', 'Karnataka', 0, NULL, NULL),

(2, 'Vikram', 'Singh', 'vikram.singh@law.edu', '9876543211', 'Model Town, Delhi', 'India', 'Delhi', 0, NULL, NULL),

(3, 'Anjali', 'Desai', 'anjali.desai.ret@email.com', '9876543212', 'Malabar Hill, Mumbai', 'India', 'Maharashtra', 0, NULL, NULL),

(4, 'Raj', 'Patel', 'raj.patel.biz@email.com', '9876543213', 'Navrangpura, Ahmedabad', 'India', 'Gujarat', 0, NULL, NULL),

(5, 'Kabir', 'Verma', 'kabir.sarcasm@email.com', '9876543214', 'Koregaon Park, Pune', 'India', 'Maharashtra', 0, NULL, NULL),

(6, 'Rohan', 'Gupta', 'rohan.g@innovatenow.com', '9876543215', 'Hitech City, Hyderabad', 'India', 'Telangana', 1, 'InnovateNow Tech Pvt. Ltd.', 'Technology'),

(7, 'Sunita', 'Krishnan', 'sunita.k@veritaslegal.com', '9876543216', 'Nariman Point, Mumbai', 'India', 'Maharashtra', 1, 'Veritas Legal LLP', 'Finance'),

(8, 'Arjun', 'Menon', 'arjun.m@greenfuture.org', '9876543217', 'Besant Nagar, Chennai', 'India', 'Tamil Nadu', 1, 'Green Future Foundation', 'Non-Profit'),

(9, 'Amitabh', 'Chaudhary', 'ac@bigcorp.com', '9876543218', 'Gurgaon, Haryana', 'India', 'Haryana', 1, 'MegaCorp Industries', 'Finance'),

(10, 'Siddharth', 'Jain', 'sid@fintechstart.com', '9876543219', 'HSR Layout, Bangalore', 'India', 'Karnataka', 1, 'Fintech Innovations Ltd', 'Technology');

INSERT INTO submissions (submission_id, user_id, draft_id) VALUES

(101, 1, 1), (102, 2, 1), (103, 3, 2), (104, 4, 2), (105, 5, 1), (106, 6, 1), (107, 7, 1), (108, 8, 3), (109, 9, 2), (110, 7, 2), (111, 4, 3), (112, 1, 2), (113, 8, 1), (114, 10, 1), (115, 5, 2);

INSERT INTO comments (submission_id, section_id, action_type, comment_text) VALUES

(101, 1, 'Suggest modification', 'While the principle of data sovereignty is important, mandating exclusive storage within India is technically challenging and financially burdensome for startups. This will create data silos and hinder access to global cloud services that offer superior security and scalability. A more pragmatic approach would be to mandate a copy of the data to be stored in India, while allowing processing on global infrastructure. This achieves security without crippling innovation.'),

(101, 3, 'Suggest modification', 'Section 22 is a necessary step, but the term "high-risk AI" is dangerously ambiguous. The draft must provide a concrete, annexed list of what constitutes critical sectors and high-risk applications. Without this clarity, the regulatory body could wield arbitrary power, creating an environment of uncertainty that would deter research and development in the AI space. The audit process must also be transparent and well-defined.'),

(102, 2, 'Suggest removal', 'This section''s attempt to dilute the safe harbour provisions contravenes established global legal principles and risks creating a chilling effect on free speech. Making platforms liable for third-party content without proof of actual knowledge or malicious intent, as established in the Shreya Singhal judgment, is legally unsustainable. This will force platforms to engage in pre-emptive censorship just to avoid litigation, which is a detriment to public discourse.'),

(105, 1, 'In Agreement', 'Oh, absolutely, data localisation is a fantastic idea! My vacation photos are surely a matter of national security and must be protected from foreign eyes at all costs. I feel so much safer knowing my data will be stored on a local server, which will undoubtedly be just as secure and efficient as the multi-billion dollar infrastructures run by global tech giants.

It’s not like India has ever had issues with infrastructure management. A truly visionary move for a self-reliant digital future.'),

(105, 3, 'In Agreement', 'A government body to regulate AI? Wonderful! I''m sure a committee of lifelong bureaucrats will have the nuanced technical expertise to fairly audit complex neural networks. They will certainly make unbiased, lightning-fast decisions that foster innovation. I eagerly await the simplified, 300-page form we''ll need to fill out in triplicate to get our new AI-powered toaster approved. This will definitely help us compete with Silicon Valley.'),

(107, 1, 'Suggest modification', 'We submit that the mandate for exclusive data storage under Section 5 imposes a disproportionate restriction on the freedom of trade and commerce. This provision may be ultra vires Article 19(1)(g) of the Constitution. A less intrusive measure, such as data mirroring, would suffice to meet the state''s objective of ensuring access for law enforcement agencies without imposing onerous costs and technical hurdles on data fiduciaries. We recommend a revision to this effect.'),

(107, 2, 'Suggest removal', 'We draw attention to Section 12. The proposed framework for intermediary liability appears to conflate the roles of publisher and platform. Imposing liability for content without a judicial order or clear evidence of incitement is a departure from settled law. Such a provision is vulnerable to being struck down for violating Articles 14 and 19 of the Constitution, as it encourages arbitrary censorship and lacks procedural safeguards for content creators and platforms alike.'),

(106, 1, 'Suggest removal', 'Section 5 is a startup killer. The capital expenditure required to comply with data localisation is prohibitive for early-stage companies. This creates a protectionist moat for large, established players and foreign corporations with deep pockets who can afford local data centers. This directly harms the government''s "Startup India" initiative. We strongly recommend this clause be entirely removed to ensure a level playing field for domestic innovators.'),

(106, 3, 'Suggest modification', 'While we support responsible AI, the regulatory framework in Section 22 should be a "light-touch" system focused on post-facto audits rather than pre-approval. A pre-approval system will drastically increase the time-to-market for innovative AI products. The regulator should work with the industry to create standards and best practices, not act as a gatekeeper. Let the market decide which products succeed, with the government stepping in only when harm is demonstrated.'),

(103, 4, 'In Agreement', 'As a former auditor, I wholeheartedly support Section 8. For too long, companies have paid lip service to social and environmental responsibilities. A standardized ESG reporting format will finally allow investors and the public to compare apples to apples. This brings much-needed transparency and will force management to think beyond quarterly profits.

The cost of compliance is a small price to pay for building long-term, sustainable and ethical businesses.'),

(103, 6, 'Suggest removal', 'Section 21, which caps executive compensation, is well-intentioned but misguided. This will lead to a brain drain of top talent to other countries. Companies must be free to offer competitive salaries to attract the best leaders. The issue is not the amount but the lack of transparency. The focus should be on strengthening the "say on pay" rules, giving shareholders more power to approve or reject compensation packages, rather than imposing an arbitrary government cap.'),

(104, 4, 'Suggest modification', 'I own a small listed company, and the ESG reporting requirement in Section 8 is a massive burden. I don''t have a dedicated department for this. This is another compliance nightmare that increases my costs and takes my focus away from actually running my business. Why should a company with a turnover of Rs. 501 Cr have the same reporting burden as one with Rs. 50,000 Cr? The threshold should be much higher, or the reporting format for smaller companies should be drastically simplified.'),

(109, 5, 'Suggest removal', 'The proposal in Section 15 to have 50% independent directors is operationally unworkable and counterproductive. While independence is important, a board heavily skewed towards non-executive directors who lack deep, intrinsic knowledge of the company''s operations can lead to poor strategic decision-making. The current 1/3rd rule is adequate. A better approach is to strengthen the definition of "independence" and enhance the role and power of the audit and nomination committees.'),

(109, 6, 'Suggest modification', 'A government-mandated cap on executive compensation is an overreach and interferes with the free market principles that guide private enterprise. Linking pay to the median salary is a flawed metric that doesn''t account for skill, risk, and global market rates for leadership. It will simply incentivize companies to outsource low-wage jobs to manipulate the median. Instead, the link should be to long-term performance metrics like market cap growth and shareholder return, as approved by the shareholders.'),

(115, 6, 'In Agreement', 'A cap on CEO salaries? Bravo! I’m sure this will magically solve income inequality overnight. Top executives, faced with the horror of earning only 100 times the median salary instead of 500, will surely find newfound humility and work purely for the love of the game. It’s a good thing talent is not a global commodity and that our best leaders would never dream of moving to a company in Singapore or Dubai for a better pay package. This policy has no potential downsides whatsoever.'),

(108, 7, 'Suggest modification', 'The solar subsidy structure in Section 4 is a positive start, but it disproportionately benefits wealthy individuals and large commercial entities who can afford the initial capital outlay. There is no clear provision for low-income households or housing

societies. The policy should include a zero-upfront-cost EMI model or community solar projects to ensure that the green transition is equitable and does not leave the poor behind.'),

(108, 8, 'In Agreement', 'The Carbon Tax outlined in Section 9 is a landmark, courageous step towards making polluters pay. This is the single most effective policy tool to drive industries towards cleaner technologies. We commend the government for its vision. We do suggest, however, that the revenue generated from this tax must be statutorily ring-fenced and used exclusively for funding renewable energy projects and supporting communities affected by the transition.'),

(102, 1, 'In Agreement', 'I find the principle of data localisation in Section 5 to be in perfect alignment with national interests. In an era of cyber-warfare and digital colonization, ensuring that our citizens'' data is governed by our laws, on our soil, is a non-negotiable aspect of sovereignty. The arguments about cost made by corporations are often exaggerated and overlook the long-term strategic benefits of building a robust domestic data infrastructure.'),

(104, 5, 'In Agreement', 'Having more independent directors is always a good thing for small investors like me. It reduces the chance of the founding family running the company like a personal fiefdom. Section 15 brings more accountability and ensures there is a truly objective voice in the boardroom that can question management''s decisions without fear. This is a very welcome change.'),

(111, 7, 'Suggest removal', 'As a small factory owner, I don''t understand these solar subsidies. The upfront cost is still too high, even with the discount, and the process to get the subsidy is full of red tape. The government should focus on providing reliable, cheap grid electricity instead of pushing us towards these complex solutions. This policy only seems to benefit the solar panel manufacturing companies.'),

(111, 8, 'Suggest modification', 'A carbon tax will destroy small industries. We operate on thin margins and cannot afford another tax. This will make my products more expensive than cheap imports. The government must provide significant grants and technological support for us to transition to greener methods before imposing a punitive tax. This is like asking a man to swim after tying his hands. We need help, not just penalties.'),

(112, 4, 'Suggest modification', 'The concept of ESG reporting is sound, but the format and details should not be a one-size-fits-all solution dictated by the government. The industry bodies like CII and FICCI should be allowed to develop sector-specific frameworks that are more relevant and less burdensome. Let the experts, not the bureaucrats, define what makes sense for each industry. This approach would be far more effective.'),

(110, 5, 'Suggest modification', 'Our firm opines that while the increase in the quota for independent directors under Section 15 is laudable, the definition of "independence" itself

remains weak. The draft fails to address the issue of long tenures and familial relationships with non-executive promoters. We recommend a stricter definition of independence, a mandatory cooling-off period, and a cap on the total tenure of any independent director to ensure their judgment remains uncompromised.'),

(113, 1, 'Suggest removal', 'Data localisation under Section 5 is an environmental concern. Building and maintaining massive data centers in India will require immense amounts of energy and water, putting a strain on our already scarce resources. It is more sustainable to leverage the hyper-efficient, green data centers that global cloud providers have already built in regions with cooler climates and abundant renewable energy. This policy is environmentally short-sighted.'),

(114, 2, 'Suggest modification', 'The new definition of intermediary liability in Section 12 is problematic for AI-driven platforms. Our platform uses algorithms to moderate content. Under the new draft, if the algorithm makes an error, would the company lose its safe harbour status? This ambiguity needs to be clarified. There must be a provision that distinguishes between programmatic moderation and willful negligence, otherwise it will stifle the use of AI for content safety.'),

(101, 2, 'In Agreement', 'I agree with redefining intermediary liability. For too long, large social media platforms have hidden behind "safe harbour" while profiting from the spread of misinformation and hate speech. This change rightfully places the onus on them to invest more in moderation and be accountable for the content they amplify. Freedom of speech is not the freedom to cause harm without consequence. This is a necessary move to make the internet a safer space.'),

(106, 2, 'Suggest removal', 'Section 12 is a recipe for disaster. This will force every tech platform to become a censor. Our small startup cannot afford a massive legal team to review every user comment. We will be forced to either over-censor and alienate our users, or face crippling lawsuits. This section disproportionately harms new Indian startups and creates an insurmountable barrier to entry, while large foreign companies can absorb the litigation costs.'),

(108, 9, 'Suggest removal', 'A phased implementation of carbon tax is not enough; it must be accompanied by a "Just Transition" fund. Section 9 will lead to job losses in traditional sectors like coal and thermal power. Where is the plan to retrain and reskill these workers? The revenue from the tax cannot just go to new green projects; a significant portion must be allocated to provide social security and new employment opportunities for those displaced by this policy.'),

(109, 4, 'Suggest modification', 'Our conglomerate has been voluntarily publishing a comprehensive sustainability report for over a decade. The government’s standardized ESG format in Section 8 is too rigid and focuses excessively on compliance rather than impact. It may

lead to a "tick-box" mentality. We propose that the government sets broad principles but allows companies the flexibility to report in a manner that is most relevant to their specific industry and stakeholders, using global standards like GRI or SASB.'),

(115, 4, 'Suggest modification', 'Oh, splendid, an ESG report! Another opportunity for corporate jargon artists to produce a glossy, 100-page document full of pictures of smiling children and thriving trees, all while the company’s factory quietly pollutes the river next door. This standardized format will surely put an end to greenwashing. I am certain that the numbers reported will be 100% accurate and never, ever manipulated to look good. This is a foolproof plan for corporate accountability.'),

(103, 5, 'In Agreement', 'I completely support raising the quota of independent directors to 50%. This is about balancing power. The executive directors, by nature, are focused on operations, and their perspective can be narrow. Independent directors bring diverse experience, an outside view, and a focus on long-term strategy, governance, and ethics. A 50-50 balance ensures that neither side can dominate the board and that decisions are made more democratically and with greater scrutiny.'),

(104, 6, 'Suggest modification', 'Capping executive salary is not the right way. It is simple jealousy politics. Instead, why not make the link to performance more transparent and strong? The rule should be that if the company''s share price goes down, the CEO''s salary also goes down significantly. And if shareholders vote against the salary package, it has to be changed. Let the owners of the company—the shareholders—decide the salary, not the government.'),

(110, 4, 'Suggest modification', 'With reference to Section 8, the proposed timeline for implementation of mandatory ESG reporting is overly aggressive. For companies to collect, audit, and report this data accurately, they need to establish new internal systems. A one-year transition period is insufficient. We recommend a phased rollout over three years, starting with the top 100 companies by market capitalization and then expanding the scope gradually. This will ensure higher quality of disclosures.'),

(112, 5, 'Suggest modification', 'While the intent of increasing independent directors is good, the pool of qualified, truly independent individuals in India is limited. Mandating a 50% quota under Section 15 will lead to the same set of individuals sitting on multiple boards, which defeats the purpose. This will lead to "over-boarding" and reduce their effectiveness. The focus should be on building capacity and training a new generation of independent directors before enforcing such a drastic quota.'),

(114, 1, 'Suggest modification', 'The definition of "sensitive personal data" in the context of data localisation needs to be extremely precise. As a fintech company, we handle financial data which is sensitive. But is user browsing behavior for financial products also sensitive? The

ambiguity in Section 5 could lead to over-compliance, where we are forced to store vast amounts of non-critical data locally, leading to massive, unnecessary costs. The draft must clearly define these categories.'),

(107, 3, 'In Agreement', 'Our firm concurs with the necessity of establishing a regulatory body for AI under Section 22. The potential for AI to cause widespread harm in areas like autonomous vehicles, medical diagnostics, and algorithmic trading is significant. Leaving such technologies completely unregulated would be a dereliction of the state''s duty to protect its citizens. We believe a well-defined regulatory framework will provide legal certainty and foster public trust, which is essential for the adoption of AI technologies.'),

(102, 3, 'Suggest removal', 'A pre-emptive regulatory body for AI, as suggested in Section 22, is a classic case of putting the cart before the horse. Innovation in AI happens at a blistering pace, and any government committee will be perpetually behind the curve, stifling progress with outdated rules. The UK’s model of pro-innovation, sector-specific regulation is a far better approach. We should regulate the *use-case* of AI (e.g., its use in credit scoring), not the *technology* itself. This section should be removed and rethought entirely.'),

(113, 8, 'Suggest modification', 'A carbon tax is necessary, but Section 9 unfairly clubs all industries together. A steel plant and a software company have vastly different carbon footprints and abilities to transition. The tax structure needs to be highly granulated by sector, with different rates and timelines. Applying a uniform tax will disproportionately harm our manufacturing sector, which is the backbone of our economy, while having little impact on the service industry.'),

(111, 9, 'Suggest modification', 'The mandate for EV charging stations in new buildings is a good idea, but 10% is too low to be meaningful. This feels like a token gesture. If we are serious about an EV transition, this number should be at least 25-30%. Furthermore, the government must also create provisions for retrofitting older buildings with charging infrastructure, which is a far bigger challenge. Section 14 is a timid step when a bold leap is required.'),

(101, 8, 'In Agreement', 'The introduction of a carbon tax in Section 9 is an economically and environmentally sound policy. By internalizing the external cost of pollution, it creates a powerful market-based incentive for companies to innovate and reduce their emissions. This is far more efficient than a command-and-control regulatory approach. The phased implementation will give industries adequate time to adapt.'),

(103, 9, 'Suggest removal', 'A carbon tax is a regressive tax that will be passed on to the consumers. The price of everything from cement to electricity will go up, and it will hurt the common citizen the most. Companies will just increase prices. This is not the right time to

introduce such a policy when we are already dealing with inflation. The government should provide incentives for going green, not penalties.'),

(104, 8, 'In Agreement', 'This carbon tax is a very good step. As a responsible businessman, I believe all industries must be accountable for their impact on the environment. For too long, pollution has been a cost borne by society, not by the polluter. Section 9 finally corrects this. It will be challenging initially, but it will force us all to become more efficient and innovative in the long run, which is good for the country and for our future generations.'),

(115, 8, 'Suggest modification', 'So now we add a carbon tax that makes everything more expensive for the middle class, while large corporations with smart accountants will find loopholes to avoid it. I''m sure the tax revenue will be used with 100% efficiency and will definitely not get lost in bureaucracy. It is comforting to know that my higher electricity bill is directly helping plant a tree somewhere, probably. A truly equitable way to solve a global crisis.'),

(102, 9, 'Suggest modification', 'The legality of a central government-imposed carbon tax as per Section 9 might face challenges from states, as taxation on production and land use often falls under state jurisdiction. The draft should clarify the constitutional basis for this tax and create a clear mechanism for revenue sharing with the states to avoid legal disputes. Without state cooperation, the implementation of this tax will be mired in legal battles.'),

(106, 9, 'In Agreement', 'As a tech company with a minimal carbon footprint, we fully support the carbon tax in Section 9. This policy will correctly incentivize a shift in investment from carbon-intensive old-economy sectors to modern, clean-tech industries like ours. It levels the playing field by making polluters pay for their negative externalities. This will accelerate the transition to a greener, technology-driven economy.'),

(108, 4, 'In Agreement', 'Mandatory ESG reporting is a vital tool for NGOs like ours. It provides us with the official, audited data we need to hold corporations accountable for their environmental promises and labor practices. Section 8 will bring a new era of corporate transparency and allow us to verify the claims companies make in their press releases. We strongly support its implementation without any dilution.'),

(109, 8, 'Suggest removal', 'Our company is already investing thousands of crores in green hydrogen and other renewable energy projects, driven by market demand and our own corporate strategy. The proposed carbon tax in Section 9 is an unnecessary and punitive measure that disincentivizes voluntary action. It is a blunt instrument that does not recognize the proactive steps being taken by responsible corporations. We advocate for a system of carbon credits and incentives, not a blanket tax.'),

(112, 8, 'Suggest modification', 'The proposed carbon tax in Section 9 should have a "carve-out" for export-oriented industries. If my products become more expensive due to this tax, I will no longer be competitive in the international market against countries that do not have such a tax. This will harm India''s exports. The tax should only be applicable to goods sold domestically, or the government must provide an equivalent duty drawback for exporters.');

"""

```
def setup_database():
    """
    Creates the database schema and populates it with initial data.
    """
    # Check if the database file already exists. If so, don't re-populate.
    db_exists = os.path.exists(DATABASE_FILE)
    if db_exists:
        print(f"Database file '{DATABASE_FILE}' already exists. Skipping creation and population.")
        # We will still connect to verify the schema is there.

    # Connect to the SQLite database. It will be created if it doesn't exist.
    conn = None
    try:
        conn = sqlite3.connect(DATABASE_FILE)
        cursor = conn.cursor()

        # IMPORTANT: Enable foreign key constraint enforcement
        cursor.execute("PRAGMA foreign_keys = ON;")
```

```
    # --- Create Tables ---
    print("Creating database schema...")
    cursor.execute(CREATE_DRAFTS_TABLE)
    cursor.execute(CREATE_SECTIONS_TABLE)
    cursor.execute(CREATE_USERS_TABLE)
    cursor.execute(CREATE_SUBMISSIONS_TABLE)
    cursor.execute(CREATE_COMMENTS_TABLE)
    print("Schema created successfully.")

    # --- Populate Tables (only if the database is new) ---
    if not db_exists:
        print("Populating database with initial data...")
        # executescript is used to run multiple SQL statements at once
        cursor.executescript(INSERT_DATA)
        print("Data inserted successfully.")
    else:
        # Check if comments table is empty, if so, populate.
        cursor.execute("SELECT COUNT(*) FROM comments")
        comment_count = cursor.fetchone()[0]
        if comment_count == 0:
            print("Database tables are empty. Populating with initial data...")
            cursor.executescript(INSERT_DATA)
            print("Data inserted successfully.")
        else:
            print(f"Found {comment_count} comments already in the database. No new data was
inserted.")
```

```
    # Commit the changes to the database file
    conn.commit()

    # --- Verification Step ---
    print("\nVerification:")
    cursor.execute("SELECT COUNT(*) FROM drafts")
    draft_count = cursor.fetchone()[0]
    print(f"- Found {draft_count} drafts.")

    cursor.execute("SELECT COUNT(*) FROM users")
    user_count = cursor.fetchone()[0]
    print(f"- Found {user_count} users.")

    cursor.execute("SELECT COUNT(*) FROM comments")
    comment_count = cursor.fetchone()[0]
    print(f"- Found {comment_count} comments.")

except sqlite3.Error as e:
    print(f"Database error: {e}")
finally:
    # Ensure the connection is closed no matter what
    if conn:
        conn.close()
        print("\nDatabase setup complete. Connection closed.")
```

if __name__ == '__main__':

    # This block runs when you execute the script directly

    setup_database()

“ and this is database description “Tables found in econsultation.db:

--- Table: drafts ---

Columns:

  - draft_id (INTEGER), Not Null: False, Default: None, Primary Key: True

  - title (TEXT), Not Null: True, Default: None, Primary Key: False

  - description (TEXT), Not Null: False, Default: None, Primary Key: False

  - created_at (DATETIME), Not Null: False, Default: CURRENT_TIMESTAMP, Primary Key: False

  - draft_ai_summary (TEXT), Not Null: False, Default: None, Primary Key: False

  - summary_positive (TEXT), Not Null: False, Default: None, Primary Key: False

  - summary_negative (TEXT), Not Null: False, Default: None, Primary Key: False

  - summary_neutral (TEXT), Not Null: False, Default: None, Primary Key: False

  - word_cloud_image_path (TEXT), Not Null: False, Default: None, Primary Key: False

  - wordcloud_positive_path (TEXT), Not Null: False, Default: None, Primary Key: False

  - wordcloud_negative_path (TEXT), Not Null: False, Default: None, Primary Key: False

  - wordcloud_neutral_path (TEXT), Not Null: False, Default: None, Primary Key: False

--- Table: sections ---

Columns:

  - section_id (INTEGER), Not Null: False, Default: None, Primary Key: True

  - draft_id (INTEGER), Not Null: True, Default: None, Primary Key: False

  - section_title (TEXT), Not Null: True, Default: None, Primary Key: False

  - section_content (TEXT), Not Null: False, Default: None, Primary Key: False

- section_ai_summary (TEXT), Not Null: False, Default: None, Primary Key: False
- section_ai_key_points (TEXT), Not Null: False, Default: None, Primary Key: False
- summary_positive (TEXT), Not Null: False, Default: None, Primary Key: False
- summary_negative (TEXT), Not Null: False, Default: None, Primary Key: False
- summary_neutral (TEXT), Not Null: False, Default: None, Primary Key: False
- word_cloud_image_path (TEXT), Not Null: False, Default: None, Primary Key: False
- wordcloud_positive_path (TEXT), Not Null: False, Default: None, Primary Key: False
- wordcloud_negative_path (TEXT), Not Null: False, Default: None, Primary Key: False
- wordcloud_neutral_path (TEXT), Not Null: False, Default: None, Primary Key: False

--- Table: users ---

Columns:

- user_id (INTEGER), Not Null: False, Default: None, Primary Key: True
- first_name (TEXT), Not Null: True, Default: None, Primary Key: False
- last_name (TEXT), Not Null: True, Default: None, Primary Key: False
- email (TEXT), Not Null: True, Default: None, Primary Key: False
- phone (TEXT), Not Null: False, Default: None, Primary Key: False
- address (TEXT), Not Null: False, Default: None, Primary Key: False
- country (TEXT), Not Null: False, Default: None, Primary Key: False
- state (TEXT), Not Null: False, Default: None, Primary Key: False
- is_organization (BOOLEAN), Not Null: False, Default: 0, Primary Key: False
- organization_name (TEXT), Not Null: False, Default: None, Primary Key: False
- industry (TEXT), Not Null: False, Default: None, Primary Key: False
- created_at (DATETIME), Not Null: False, Default: CURRENT_TIMESTAMP, Primary Key: False

--- Table: submissions ---

Columns:

- submission_id (INTEGER), Not Null: False, Default: None, Primary Key: True
- user_id (INTEGER), Not Null: True, Default: None, Primary Key: False
- draft_id (INTEGER), Not Null: True, Default: None, Primary Key: False
- otp_verified (BOOLEAN), Not Null: False, Default: 0, Primary Key: False
- submission_status (TEXT), Not Null: True, Default: 'completed', Primary Key: False
- submitted_at (DATETIME), Not Null: False, Default: CURRENT_TIMESTAMP, Primary Key: False

--- Table: comments ---

Columns:

- comment_id (INTEGER), Not Null: False, Default: None, Primary Key: True
- submission_id (INTEGER), Not Null: True, Default: None, Primary Key: False
- section_id (INTEGER), Not Null: True, Default: None, Primary Key: False
- action_type (TEXT), Not Null: True, Default: None, Primary Key: False
- comment_text (TEXT), Not Null: False, Default: None, Primary Key: False
- created_at (DATETIME), Not Null: False, Default: CURRENT_TIMESTAMP, Primary Key: False
- updated_at (DATETIME), Not Null: False, Default: None, Primary Key: False
- sentiment_label (TEXT), Not Null: False, Default: None, Primary Key: False
- sentiment_score (REAL), Not Null: False, Default: None, Primary Key: False
- score_positive (REAL), Not Null: False, Default: None, Primary Key: False
- score_negative (REAL), Not Null: False, Default: None, Primary Key: False
- score_neutral (REAL), Not Null: False, Default: None, Primary Key: False
- ai_summary (TEXT), Not Null: False, Default: None, Primary Key: False
- word_cloud_image_path (TEXT), Not Null: False, Default: None, Primary Key: False”

This is the layout.svelte “<main class="app-shell">

```
  <nav class="main-nav">
    <a href="/overall-analysis">Overall Analysis</a>
    <a href="/">Sentiment Analysis Dashboard</a>
    <a href="/summaries">Text Summary & Word Cloud</a>
  </nav>
  <div class="content">
    <slot />
  </div>
</main>

<style>
  :global(body) {
    font-family: system-ui, -apple-system, sans-serif;
    margin: 0;
    background-color: #f8f9fa;
    color: #212529;
  }
  .main-nav {
    background-color: white;
    padding: 0 2rem;
    display: flex;
    gap: 2rem;
    box-shadow: 0 2px 4px rgba(0,0,0,0.05);
    border-bottom: 1px solid #dee2e6;
  }
  .main-nav a {
```

```
    color: #495057;
    text-decoration: none;
    font-size: 1rem;
    font-weight: 500;
    padding: 1.5rem 0;
    border-bottom: 3px solid transparent;
  }
  .content {
    padding: 2rem;
    max-width: 1400px;
    margin: 0 auto;
  }
</style>
“ and this is summaries (text summaries and word cloud) “<script>
    import { onMount } from 'svelte';

    // --- State Management ---
    let drafts = [];
    let allComments = [];
    let sections = [];
    let selectedDraftId = '';
    let selectedSectionId = 'all';

    // Filter states for comments
    let filterState = 'All';
    let filterAction = 'All';
```

```
  let filterSentiment = 'All';

  // UI State
  let showComments = false;

  // --- Derived State ---
  $: selectedDraft = drafts.find(d => d.draft_id == selectedDraftId) || null;
  $: selectedSection = sections.find(s => s.section_id == selectedSectionId) || null;

  // Sections to display based on selection
  $: displaySections = selectedSectionId === 'all' ? sections : sections.filter(s => s.section_id == selectedSectionId);

  // Available filter options
  $: availableStates = ['All', ...new Set(allComments.map(c => c.state).filter(c => c))];
  $: availableActions = ['All', ...new Set(allComments.map(c => c.action_type).filter(c => c))];
  $: availableSentiments = ['All', ...new Set(allComments.map(c => c.sentiment_label).filter(c => c))];

  // Filtered comments based on all filters
  $: filteredComments = (() => {
    if (!selectedDraftId) return [];

    let filtered = allComments;

    if (selectedSectionId !== 'all') {
      filtered = filtered.filter(c => c.section_id == selectedSectionId);
```

```
    }

    if (filterState !== 'All') {
        filtered = filtered.filter(c => c.state == filterState);
    }

    if (filterAction !== 'All') {
        filtered = filtered.filter(c => c.action_type == filterAction);
    }

    if (filterSentiment !== 'All') {
        filtered = filtered.filter(c => c.sentiment_label == filterSentiment);
    }

    return filtered;
})();

// Reset filters when draft or section changes
$: if (selectedDraftId || selectedSectionId) {
    resetFilters();
}

function resetFilters() {
    filterState = 'All';
    filterAction = 'All';
    filterSentiment = 'All';
```

```
}

const API_BASE_URL = 'http://127.0.0.1:5000';

onMount(async () => {
    try {
        const res = await fetch(`${API_BASE_URL}/api/drafts`);
        drafts = await res.json();
    } catch (error) {
        console.error('Error fetching drafts:', error);
    }
});

// Handle draft change
$: if (selectedDraftId) handleDraftChange();

async function handleDraftChange() {
    if (!selectedDraftId) {
        sections = [];
        allComments = [];
        showComments = true;
        return;
    }
    selectedSectionId = 'all';
    showComments = true;
```

```
        try {
            const [sectionsRes, commentsRes] = await Promise.all([
                fetch(`${API_BASE_URL}/api/sections/${selectedDraftId}`),
                fetch(`${API_BASE_URL}/api/comments/${selectedDraftId}`)
            ]);
            sections = await sectionsRes.json();
            allComments = await commentsRes.json();
        } catch (error) {
            console.error('Error fetching data:', error);
        }
    }

    function getSentimentColor(label) {
        if (label === 'Positive') return '#28a745';
        if (label === 'Negative') return '#dc3545';
        return '#6c757d';
    }

    // function toggleComments() {
    //     showComments = !showComments;
    // }
</script>

<div class="container">
```

```
<!-- Step 1: Draft Selection -->
<section class="section draft-selection">

  <div class="draft-selector">
    <select bind:value={selectedDraftId} class="primary-select">
      <option value="">-- Draft Selection --</option>
      {#each drafts as draft}
        <option value={draft.draft_id}>{draft.title}</option>
      {/each}
    </select>
  </div>
</section>

{#if selectedDraft}
  <!-- Step 2: Overview Section -->
  <section class="section overview-section">
    <div class="section-header">
      <h2>Draft Overview</h2>

    </div>

    <div class="overview-grid">
      <!-- Word Cloud -->
      <div class="card wordcloud-card">
        <div class="card-header">
```

```
            <h3>Draft Word Cloud</h3>
            <p class="card-subtitle">Most frequently mentioned terms</p>
          </div>
          <div class="card-body">
            {#if selectedDraft.word_cloud_image_path}
              <img src="{API_BASE_URL}/{selectedDraft.word_cloud_image_path}" alt="Word
Cloud">
            {:else}
              <p class="empty-state">No word cloud available</p>
            {/if}
          </div>
        </div>

        <!-- Summary -->
        <div class="card summary-card">
          <div class="card-header">
            <h3>Summary of Comments recevied for the Draft</h3>
            <p class="card-subtitle">AI-generated overview of feedback</p>
          </div>
          <div class="card-body">
            <div class="summary-text">
              {selectedDraft.draft_ai_summary || 'No summary available for this draft.'}
            </div>
          </div>
        </div>
      </div>
```

```
      </section>

      <!-- Step 3: Section Summaries -->
<!-- Step 3: Section Summaries -->
{#if displaySections.length > 0}
   <section class="section sections-section">
      <div class="section-header">
         <h2>
            {selectedSectionId === 'all' ? 'Section Summaries' : 'Selected Section'}
         </h2>
         <!-- <div class="section-meta">
            {displaySections.length} {displaySections.length === 1 ? 'section' : 'sections'}
         </div> -->
         <div class="section-controls">
            <label>
               <select bind:value={selectedSectionId} class="inline-select">
                  <option value="all">--Select a section to view Section Word Cloud--</option>
                  {#each sections as section}
                     <option value={section.section_id}>{section.section_title}</option>
                  {/each}
               </select>
            </label>
         </div>
      </div>

      {#if selectedSectionId !== 'all'}
```

```
        <!-- Side-by-side layout for single section -->
        <div class="section-detail-grid">
          <!-- Word Cloud -->
          <div class="card section-wordcloud-card">
            <div class="card-header">
              <h3>Section Word Cloud</h3>
              <p class="card-subtitle">Key terms in this section</p>
            </div>
            <div class="card-body wordcloud-body">
              {#if selectedSection?.word_cloud_image_path}
                <img src="{API_BASE_URL}/{selectedSection.word_cloud_image_path}"
alt="Section Word Cloud">
              {:else}
                <p class="empty-state">No word cloud available for this section</p>
              {/if}
            </div>
          </div>

          <!-- Section Summary -->
          <div class="card section-card">
            <div class="card-header">
              <h3>{selectedSection?.section_title}</h3>
            </div>
            <div class="card-body">
              <div class="section-content">
                <div class="summary-block">
```

```
                <strong>Summary:</strong>
                <p>{selectedSection?.section_ai_summary || 'No summary available.'}</p>
              </div>
              {#if selectedSection?.section_ai_key_points}
                <div class="keypoints-block">
                  <strong>Key Points:</strong>
                  <p>{selectedSection.section_ai_key_points}</p>
                </div>
              {/if}
            </div>
          </div>
        </div>
      </div>
    {:else}
      <!-- Grid layout for all sections -->
      <div class="sections-grid">
        {#each displaySections as section}
          <div class="card section-card">
            <div class="card-header">
              <h3>{section.section_title}</h3>
            </div>
            <div class="card-body">
              <div class="section-content">
                <div class="summary-block">
                  <strong>Summary:</strong>
                  <p>{section.section_ai_summary || 'No summary available.'}</p>
```

```
                    </div>
                    {#if section.section_ai_key_points}
                      <div class="keypoints-block">
                        <strong>Key Points:</strong>
                        <p>{section.section_ai_key_points}</p>
                      </div>
                    {/if}
                  </div>
                </div>
              </div>
            {/each}
          </div>
        {/if}
      </section>
{/if}

      <!-- Step 4: Comments Section -->
      <section class="section comments-section">
        <div class="section-header">
          <h2>Individual Comments</h2>

        </div>

        {#if showComments}
          <div class="filters-panel">
            <div class="filters-header">
```

```
    <span class="filters-label">Filter Comments:</span>
  </div>
  <div class="filters-grid">
    <label class="filter-group">
      <span>State</span>
      <select bind:value={filterState}>
        {#each availableStates as state}
          <option value={state}>{state}</option>
        {/each}
      </select>
    </label>

    <label class="filter-group">
      <span>Action Type</span>
      <select bind:value={filterAction}>
        {#each availableActions as action}
          <option value={action}>{action}</option>
        {/each}
      </select>
    </label>

    <label class="filter-group">
      <span>Sentiment</span>
      <select bind:value={filterSentiment}>
        {#each availableSentiments as sentiment}
          <option value={sentiment}>{sentiment}</option>
```

```
                {/each}
              </select>
            </label>
          </div>
        </div>

        <div class="card comments-card">
          <div class="comments-list">
            {#if filteredComments.length === 0}
              <div class="empty-state">
                <p>No comments match the selected filters.</p>
                <button class="link-btn" on:click={resetFilters}>Clear all filters</button>
              </div>
            {:else}
              {#each filteredComments as comment, i}
                <div class="comment-item">
                  <div class="comment-header">
                    <span class="comment-number">#{i + 1}</span>
                    <div class="comment-badges">
                      <span
                        class="badge sentiment-badge"
                        style:background-
color={getSentimentColor(comment.sentiment_label)}
                      >
                        {comment.sentiment_label || 'N/A'}
                      </span>
```

```
                        <span class="badge action-badge">
                            {comment.action_type}
                        </span>
                    </div>
                </div>

                <div class="comment-content-grid">
                    <!-- Original Comment -->
                    <div class="comment-block original-comment">
                        <div class="block-header">
                            <svg width="16" height="16" viewBox="0 0 16 16"
fill="currentColor">
                                <path d="M2.5 3A1.5 1.5 0 0 0 1
4.5v.793c.026.009.051.02.076.032L7.674 8.51c.206.1.446.1.652 0l6.598-3.185A.755.755 0 0 1
15 5.293V4.5A1.5 1.5 0 0 0 13.5 3h-11Z"/>
                                <path d="M15 6.954 8.978 9.86a2.25 2.25 0 0 1-1.956 0L1
6.954V11.5A1.5 1.5 0 0 0 2.5 13h11a1.5 1.5 0 0 0 1.5-1.5V6.954Z"/>
                            </svg>
                            Original Comment
                        </div>
                        <p class="comment-text">{comment.comment_text || 'No comment
text available'}</p>
                    </div>

                    <!-- AI Summary -->
                    <div class="comment-block ai-summary">
                        <div class="block-header">
```

<svg width="16" height="16" viewBox="0 0 16 16" fill="currentColor">

<path d="M8 0a8 8 0 1 0 0 16A8 8 0 0 0 8 0zm.93 4.588l-2.29.287-.082.38.45.083c.294.07.352.176.288.469l-.738 3.468c-.194.897.105 1.319.808 1.319.545 0 1.178-.252 1.465-.598l.088-.416c-.2.176-.492.246-.686.246-.275 0-.375-.193-.304-.533L8.93 4.588zM9 3.5a1 1 0 1 1-2 0 1 1 0 0 1 2 0z"/>

</svg>

AI Summary

{comment.ai_summary || 'No AI summary available'}

<!-- Metadata -->

{#if comment.section_title}

Section:

{comment.section_title}

{/if}

{#if comment.state}

State:

{comment.state}

```
                        {/if}
                        {#if comment.industry}
                          <span class="meta-item">
                            <span class="meta-label">Industry:</span>
                            <span class="meta-value">{comment.industry}</span>
                          </span>
                        {/if}
                      </div>
                    </div>
                  {/each}
                {/if}
              </div>
            </div>
          {/if}
        </section>
      {:else}
        <div class="empty-state-large">
          <svg width="64" height="64" viewBox="0 0 24 24" fill="none" stroke="currentColor" stroke-width="2">
            <path d="M14 2H6a2 2 0 0 0-2 2v16a2 2 0 0 0 2 2h12a2 2 0 0 0 2-2V8z"/>
            <polyline points="14 2 14 8 20 8"/>
            <line x1="16" y1="13" x2="8" y2="13"/>
            <line x1="16" y1="17" x2="8" y2="17"/>
            <polyline points="10 9 9 9 8 9"/>
          </svg>
          <h3>No Draft Selected</h3>
```

```
            <p>Select a draft from the dropdown above to view its analysis and feedback</p>
        </div>
    {/if}
</div>

<style>
    :global(body) {
        margin: 0;
        font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, 'Helvetica Neue', Arial, sans-serif;
        background: #f5f7fa;
        color: #1a202c;
    }

    .container {
        max-width: 1400px;
        margin: 0 auto;
        padding: 1rem;
    }

    /* Header */
```

```
/* Sections */
.section {
    margin-bottom: 2.5rem;
}

.section-header {
    display: flex;
    justify-content:center;
    align-items: center;
    margin-bottom: 1.5rem;
    padding-bottom: 0.75rem;
    border-bottom: 2px solid #e2e8f0;
    gap:10px;

}

.section-header h2 {
    margin: 0;
    font-size: 1.5rem;
    font-weight: 600;
    color: #2d3748;
    text-align: center;
}

.section-meta {
```

```
    font-size: 0.9rem;
    color: #718096;
    font-weight: 500;
}

.section-controls {
    display: flex;
    align-items: center;
    gap: 1rem;
}

.label-text {
    font-size: 0.9rem;
    color: #4a5568;
    margin-right: 0.5rem;
}

/* Draft Selection */
.draft-selector {
    display: flex;
    justify-content: center;
}

.primary-select {
    font-size: 1.1rem;
    padding: 1rem 1.5rem;
```

```
    border-radius: 12px;
    border: 2px solid #cbd5e0;
    background: white;
    cursor: pointer;
    min-width: 500px;
    transition: all 0.2s;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}

.primary-select:hover {
    border-color: #4299e1;
}

.primary-select:focus {
    outline: none;
    border-color: #3182ce;
    box-shadow: 0 0 0 3px rgba(66, 153, 225, 0.1);
}

.inline-select {
    font-size: 0.95rem;
    padding: 0.5rem 0.75rem;
    border-radius: 8px;
    border: 1px solid #cbd5e0;
    background: white;
    cursor: pointer;
```

```
    transition: all 0.2s;
}

.inline-select:hover {
    border-color: #a0aec0;
}

.inline-select:focus {
    outline: none;
    border-color: #3182ce;
}

/* Cards */
.card {
    background: white;
    border-radius: 12px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
    overflow: hidden;
}

.card-header {
    padding: 1.5rem;
    border-bottom: 1px solid #e2e8f0;
}

.card-header h3 {
```

```
    margin: 0 0 0.25rem 0;
    font-size: 1.1rem;
    font-weight: 600;
    color: #2d3748;
}

.card-subtitle {
    margin: 0;
    font-size: 0.875rem;
    color: #718096;
}

.card-body {
    padding: 1.5rem;
}

/* Overview Grid */
.overview-grid {
    display: grid;
    grid-template-columns: 1fr 2fr;
    gap: 1.5rem;
}

.wordcloud-card .card-body {
    display: flex;
    align-items: center;
```

```
    justify-content: center;
    min-height: 300px;
  }

  .wordcloud-card img {
    max-width: 100%;
    max-height: 350px;
    object-fit: contain;
  }

  .summary-text {
    line-height: 1.8;
    color: #4a5568;
    font-size: 1rem;
  }

  /* Sections Grid */
  .sections-grid {
    display: grid;
    grid-template-columns: repeat(auto-fill, minmax(400px, 1fr));
    gap: 1.5rem;
  }
  /* Section Detail Grid - Side by Side Layout */
.section-detail-grid {
  display: grid;
  grid-template-columns: 1fr 1fr;
```

```
    gap: 1.5rem;
}

.section-wordcloud-card .card-body {
    display: flex;
    align-items: center;
    justify-content: center;
    min-height: 400px;
}

.section-wordcloud-card img {
    max-width: 100%;
    max-height: 450px;
    object-fit: contain;
}

@media (max-width: 1200px) {
    .section-detail-grid {
        grid-template-columns: 1fr;
    }
}

    .section-content {
        display: flex;
        flex-direction: column;
        gap: 1rem;
```

```
}

.summary-block,
.keypoints-block {
    padding: 0;
}

.summary-block strong,
.keypoints-block strong {
    display: block;
    margin-bottom: 0.5rem;
    color: #2d3748;
    font-size: 0.9rem;
    text-transform: uppercase;
    letter-spacing: 0.5px;
}

.summary-block p,
.keypoints-block p {
    margin: 0;
    color: #4a5568;
    line-height: 1.6;
}

.keypoints-block {
    padding-top: 1rem;
```

```
    border-top: 1px solid #e2e8f0;
}

/* Filters Panel */
.filters-panel {
    background: white;
    border-radius: 12px;
    padding: 1.5rem;
    margin-bottom: 1.5rem;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}

.filters-header {
    margin-bottom: 1rem;
}

.filters-label {
    font-size: 0.9rem;
    font-weight: 600;
    color: #2d3748;
    text-transform: uppercase;
    letter-spacing: 0.5px;
}

.filters-grid {
    display: grid;
```

```
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 1rem;
}

.filter-group {
    display: flex;
    flex-direction: column;
    gap: 0.5rem;
}

.filter-group span {
    font-size: 0.875rem;
    font-weight: 500;
    color: #4a5568;
}

.filter-group select {
    padding: 0.5rem 0.75rem;
    border-radius: 6px;
    border: 1px solid #cbd5e0;
    background: white;
    cursor: pointer;
    transition: all 0.2s;
}

.filter-group select:hover {
```

```
        border-color: #a0aec0;
    }

    .filter-group select:focus {
        outline: none;
        border-color: #3182ce;
    }

    /* Comments */
    .comments-list {
    padding: 1rem;
}

.comment-item {
    border: 1px solid #e2e8f0;
    border-radius: 8px;
    padding: 1.5rem;
    margin-bottom: 1.5rem;
    background: white;
}

.comment-item:last-child {
    margin-bottom: 0;
}
```

```
    .comment-header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    margin-bottom: 1.25rem;
    padding-bottom: 0.75rem;
    border-bottom: 2px solid #e2e8f0;
}

.comment-number {
    font-weight: 700;
    color: #2d3748;
    font-size: 1.1rem;
}

.comment-badges {
    display: flex;
    gap: 0.5rem;
}

.badge {
    font-size: 0.75rem;
    font-weight: 600;
    padding: 0.25rem 0.75rem;
    border-radius: 12px;
```

```
    text-transform: uppercase;
    letter-spacing: 0.5px;
}

.sentiment-badge {
    color: white;
}

.action-badge {
    background: #edf2f7;
    color: #4a5568;
}

.comment-content-grid {
    display: grid;
    grid-template-columns: 1fr 1fr;
    gap: 1.5rem;
    margin-bottom: 1rem;
}

.comment-block {
    padding: 1rem;
    border-radius: 6px;
    background: #f7fafc;
}
```

```
.original-comment {
    border-left: 3px solid #718096;
}

.ai-summary {
    border-left: 3px solid #4299e1;
}

.block-header {
    display: flex;
    align-items: center;
    gap: 0.5rem;
    font-weight: 600;
    color: #2d3748;
    font-size: 0.9rem;
    margin-bottom: 0.75rem;
}

.block-header svg {
    flex-shrink: 0;
}

.original-comment .block-header svg {
    color: #718096;
```

```
}

.ai-summary .block-header svg {
    color: #4299e1;
}

.comment-text {
    margin: 0;
    color: #4a5568;
    line-height: 1.7;
    font-size: 0.95rem;
}

.comment-metadata {
    display: flex;
    flex-wrap: wrap;
    gap: 1.5rem;
    padding-top: 1rem;
    border-top: 1px solid #e2e8f0;
}

.meta-item {
    display: flex;
    align-items: center;
    gap: 0.5rem;
}
```

```
.meta-label {
    font-size: 0.875rem;
    color: #718096;
    font-weight: 500;
}

.meta-value {
    font-size: 0.875rem;
    color: #2d3748;
    font-weight: 600;
}

@media (max-width: 968px) {
    .comment-content-grid {
        grid-template-columns: 1fr;
    }
}

@media (max-width: 768px) {


}

    .toggle-btn:hover {
        background: #2c5aa0;
    }
```

```
.link-btn {
    background: none;
    border: none;
    color: #3182ce;
    cursor: pointer;
    text-decoration: underline;
    padding: 0;
    font-size: 0.95rem;
}

.link-btn:hover {
    color: #2c5aa0;
}

/* Empty States */
.empty-state {
    text-align: center;
    padding: 3rem 2rem;
    color: #718096;
}

.empty-state-large {
    text-align: center;
    padding: 5rem 2rem;
    background: white;
```

```
    border-radius: 12px;
    box-shadow: 0 1px 3px rgba(0,0,0,0.1);
}

.empty-state-large svg {
    color: #cbd5e0;
    margin-bottom: 1.5rem;
}

.empty-state-large h3 {
    margin: 0 0 0.5rem 0;
    color: #2d3748;
    font-size: 1.5rem;
}

.empty-state-large p {
    margin: 0;
    color: #718096;
    font-size: 1rem;
}

/* Responsive */
@media (max-width: 1200px) {
    .overview-grid {
        grid-template-columns: 1fr;
    }
```

```
    .sections-grid {
        grid-template-columns: 1fr;
    }
}

@media (max-width: 768px) {
    .container {
        padding: 1rem;
    }

    .page-header h1 {
        font-size: 1.75rem;
    }

    .primary-select {
        min-width: 100%;
    }

    .section-header {
        flex-direction: column;
        align-items: flex-start;
        gap: 1rem;
    }

    .filters-grid {
```

```
            grid-template-columns: 1fr;
        }
    }
</style>
“
```