Implementation

To explore these ideas in practice, I built a small sentiment analyzer using Laravel, Prism PHP, and an OpenAI-compatible model. For each piece of feedback, it returns a sentiment label, a confidence score, key topics, and a one-sentence summary. The goal wasn't to build a production tool, but to observe where AI sentiment analysis performs well - and where its limitations become apparent.

Tech stack used: PHP 8.2, Laravel 12, Prism PHP v0.100.1

Commands ran:

- Create Laravel project:

```
composer create-project laravel/laravel sentiment-analyzer
```

- Get Prism PHP:

```
composer require prism-php/prism
php artisan vendor:publish --tag=prism-config
```

Configuration Setup

- .env file:

  Add variables like PROVIDER_NAME, PROVIDER_API_KEY, PROVIDER_URL, and PROVIDER_MODEL with the values for whichever provider you're using - OpenAI, Anthropic, or another compatible provider.

- config/prism.php:

  Add those env variables to the provider whose credentials you have. For example, with OpenAI credentials, update the file like this:

```
'openai' => [
    'provider_name' => env('PROVIDER_NAME', 'openai'),
    'url' => env('PROVIDER_URL', 'https://api.openai.com/v1'),
    'api_key' => env('PROVIDER_API_KEY', ''),
    'model' => env('PROVIDER_MODEL'),
],
```

Code

Controller: SentimentController.php

```
<?php

namespace App\Http\Controllers;

use App\Services\SentimentAnalyzer;
use Illuminate\Http\Request;
```

```
class SentimentController extends Controller
{
    public function index()
    {
        return view('sentiment');
    }

    public function analyze(Request $request, SentimentAnalyzer $analyzer)
    {
        $request->validate([
            'text' => 'required|string|min:3|max:1000',
        ]);

        try {
            $result = $analyzer->analyze($request->input('text'));

            return response()->json([
                'success' => true,
                'data' => $result,
            ]);
        } catch (\Exception $e) {
            return response()->json([
                'success' => false,
                'message' => $e->getMessage(),
            ], 500);
        }
    }
}
```

Service Class: SentimentAnalyzer.php

```
<?php

namespace App\Services;

use Prism\Prism\Facades\Prism;
use Prism\Prism\ValueObjects\Messages\UserMessage;
use Illuminate\Support\Facades\Log;

class SentimentAnalyzer
{
    /**
     * Analyze sentiment of the given text.
     *
     * @param string $text
     * @return array{ sentiment: string, confidence: float, topics: array, summary:
string }
     * @throws \Exception
     */
    public function analyze(string $text): array
    {
        $prompt = <<<PROMPT
        You are a sentiment analysis expert. Analyze the following text and return a JSON
```

```
response with:
        - "sentiment": one of "positive", "negative", "neutral", "mixed", or "sarcastic".
        Definitions:
        - "positive": clear favorable tone.
        - "negative": clear unfavorable tone.
        - "neutral": factual, no strong emotion.
        - "mixed": contains both clear positive and clear negative elements.
        - "sarcastic": says the opposite of what is meant, often mocking or ironic.
        - "confidence": a number between 0 and 1
        - "topics": an array of key topics mentioned (up to 5)
        - "summary": a brief one-sentence summary

        Text: "$text"
        PROMPT;

        try {
            // Call the model via Prism
            $response = Prism::text()
                ->using(config('prism.providers.openai.provider_name'),
config('prism.providers.openai.model'))
                ->withPrompt($prompt)
                ->withProviderOptions([
                    'reasoning' => ['effort' => 'high'],
                ])
                ->withMaxTokens(4096)
                ->generate();

            // Get the raw text from the response
            $raw = $response->text;

            // Parse JSON from the response
            $data = $this->parseJsonResponse($raw);

            // Validate and return structured array
            return $this->validateAndStructure($data);

        } catch (\Exception $e) {
            Log::error('Sentiment analysis failed: ' . $e->getMessage());
            throw new \Exception('AI service error: ' . $e->getMessage());
        }
    }

    /**
     * Extract JSON from the model's response (robust against extra text).
     */
    private function parseJsonResponse(string $raw): array
    {
        // Attempt to find JSON block between ```json ... ``` or just raw JSON
        if (preg_match('/```json\s*([\s\S]*?)\s*```/', $raw, $matches)) {
            $json = $matches[1];
        } elseif (preg_match('/\{[\s\S]*\}/', $raw, $matches)) {
            $json = $matches[0];
        } else {
            throw new \Exception('No valid JSON found in AI response.');
        }
```

```
        $data = json_decode($json, true);
        if (json_last_error() !== JSON_ERROR_NONE) {
            throw new \Exception('Failed to parse JSON: ' . json_last_error_msg());
        }

        return $data;
    }

    /**
     * Ensure the required fields exist and have correct types.
     */
    private function validateAndStructure(array $data): array
    {
        $sentiment = $data['sentiment'] ?? 'neutral';
        if (!in_array($sentiment, ['positive', 'negative', 'neutral', 'mixed',
'sarcastic'])) {
            $sentiment = 'neutral';
        }

        $confidence = (float) ($data['confidence'] ?? 0.5);
        $confidence = max(0, min(1, $confidence));

        $topics = (array) ($data['topics'] ?? []);
        $topics = array_slice($topics, 0, 5);

        $summary = (string) ($data['summary'] ?? 'No summary provided.');

        return [
            'sentiment' => $sentiment,
            'confidence' => $confidence,
            'topics' => $topics,
            'summary' => $summary,
        ];
    }
}
```

Blade file: sentiment.blade.php

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Sentiment Analyzer</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            background: #f7f9fc;
            display: flex;
```

```
        justify-content: center;
        align-items: center;
        min-height: 100vh;
        padding: 20px;
    }

    .card {
        background: white;
        border-radius: 20px;
        padding: 40px;
        max-width: 700px;
        width: 100%;
        box-shadow: 0 10px 30px rgba(0,0,0,0.1);
    }

    h1 {
        font-size: 28px;
        color: #1a202c;
        margin-bottom: 8px;
    }

    .subtitle {
        color: #718096;
        margin-bottom: 30px;
        font-size: 16px;
    }

    textarea {
        width: 100%;
        padding: 14px 18px;
        border: 2px solid #e2e8f0;
        border-radius: 12px;
        font-size: 16px;
        font-family: inherit;
        resize: vertical;
        transition: border-color 0.2s;
        min-height: 120px;
    }

    textarea:focus {
        outline: none;
        border-color: #5a67d8;
        box-shadow: 0 0 0 3px rgba(90, 103, 216, 0.2);
    }

    .button {
        margin-top: 16px;
        background: #5a67d8;
        color: white;
        border: none;
        padding: 12px 28px;
        border-radius: 30px;
        font-size: 16px;
        font-weight: 600;
        cursor: pointer;
        transition: background 0.2s, transform 0.1s;
    }

    .button:hover {
        background: #4c51bf;
    }
```

```
.button:active {
    transform: scale(0.97);
}

.button:disabled {
    opacity: 0.6;
    cursor: not-allowed;
}

.loader {
    display: none;
    margin-left: 12px;
    border: 3px solid #e2e8f0;
    border-top: 3px solid #5a67d8;
    border-radius: 50%;
    width: 20px;
    height: 20px;
    animation: spin 0.8s linear infinite;
    vertical-align: middle;
}

@keyframes spin {
    to { transform: rotate(360deg); }
}

.result {
    margin-top: 30px;
    padding-top: 20px;
    border-top: 2px solid #edf2f7;
    display: none;
}

.result.visible {
    display: block;
}

.sentiment-header {
    display: flex;
    align-items: center;
    gap: 12px;
    margin-bottom: 14px;
}

.emoji {
    font-size: 42px;
    line-height: 1;
}

.sentiment-label {
    font-size: 22px;
    font-weight: 600;
    text-transform: capitalize;
}

.sentiment-positive .sentiment-label { color: #38a169; }
.sentiment-negative .sentiment-label { color: #e53e3e; }
.sentiment-neutral .sentiment-label { color: #d69e2e; }
.sentiment-mixed .sentiment-label { color: #ed8936; }
.sentiment-sarcastic .sentiment-label { color: #805ad5; }
```

```
        .summary {
            font-size: 18px;
            line-height: 1.6;
            color: #2d3748;
            background: #f7fafc;
            padding: 16px 20px;
            border-radius: 12px;
            margin: 12px 0 18px;
        }

        .meta {
            display: flex;
            flex-wrap: wrap;
            gap: 20px;
            font-size: 15px;
            color: #4a5568;
        }

        .meta-item {
            background: #edf2f7;
            padding: 4px 14px;
            border-radius: 20px;
        }

        .topics {
            display: flex;
            flex-wrap: wrap;
            gap: 8px;
            margin-top: 12px;
        }

        .topic-tag {
            background: #e2e8f0;
            padding: 4px 14px;
            border-radius: 20px;
            font-size: 14px;
            color: #2d3748;
        }

        .error {
            margin-top: 20px;
            padding: 12px 18px;
            background: #fed7d7;
            color: #c53030;
            border-radius: 10px;
            display: none;
        }

        .error.visible {
            display: block;
        }

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

        @media (max-width: 500px) {
            .card { padding: 24px; }
```

```
            h1 { font-size: 22px; }
        }
    </style>
</head>
<body>
    <div class="card">
        <h1>&#129504; Sentiment Analyzer</h1>
        <p class="subtitle">Paste a review or feedback -- we'll tell you the mood.</p>

        <form id="analyzeForm">
            @csrf
            <textarea id="textInput" placeholder="E.g., This product exceeded all my expectations! The
customer service was outstanding." required></textarea>
            <div class="flex-row">
                <button type="submit" class="button" id="submitBtn">
                    <span id="btnText">Analyze</span>
                    <span class="loader" id="loader"></span>
                </button>
            </div>
        </form>

        <div id="error" class="error"></div>

        <div id="result" class="result">
            <div class="sentiment-header" id="sentimentHeader">
                <span class="emoji" id="emoji"></span>
                <span class="sentiment-label" id="sentimentLabel"></span>
                <span class="meta-item" id="confidenceBadge">Confidence: <span
id="confidenceValue"></span>%</span>
            </div>
            <div class="summary" id="summaryText"></div>
            <div class="meta">
                <span class="meta-item">&#128204; Topics</span>
            </div>
            <div class="topics" id="topicsContainer"></div>
        </div>
    </div>

    <script>
        (function() {
            const form = document.getElementById('analyzeForm');
            const textInput = document.getElementById('textInput');
            const submitBtn = document.getElementById('submitBtn');
            const btnText = document.getElementById('btnText');
            const loader = document.getElementById('loader');
            const errorDiv = document.getElementById('error');
            const resultDiv = document.getElementById('result');
            const emojiSpan = document.getElementById('emoji');
            const sentimentLabel = document.getElementById('sentimentLabel');
            const confidenceValue = document.getElementById('confidenceValue');
            const summaryText = document.getElementById('summaryText');
            const topicsContainer = document.getElementById('topicsContainer');
            const sentimentHeader = document.getElementById('sentimentHeader');

            // Emoji mapping
            const emojiMap = {
                positive: '😊',
                negative: '😞',
                neutral: '😐',
                mixed: '🤔',
                sarcastic: '🙃'
            };
```

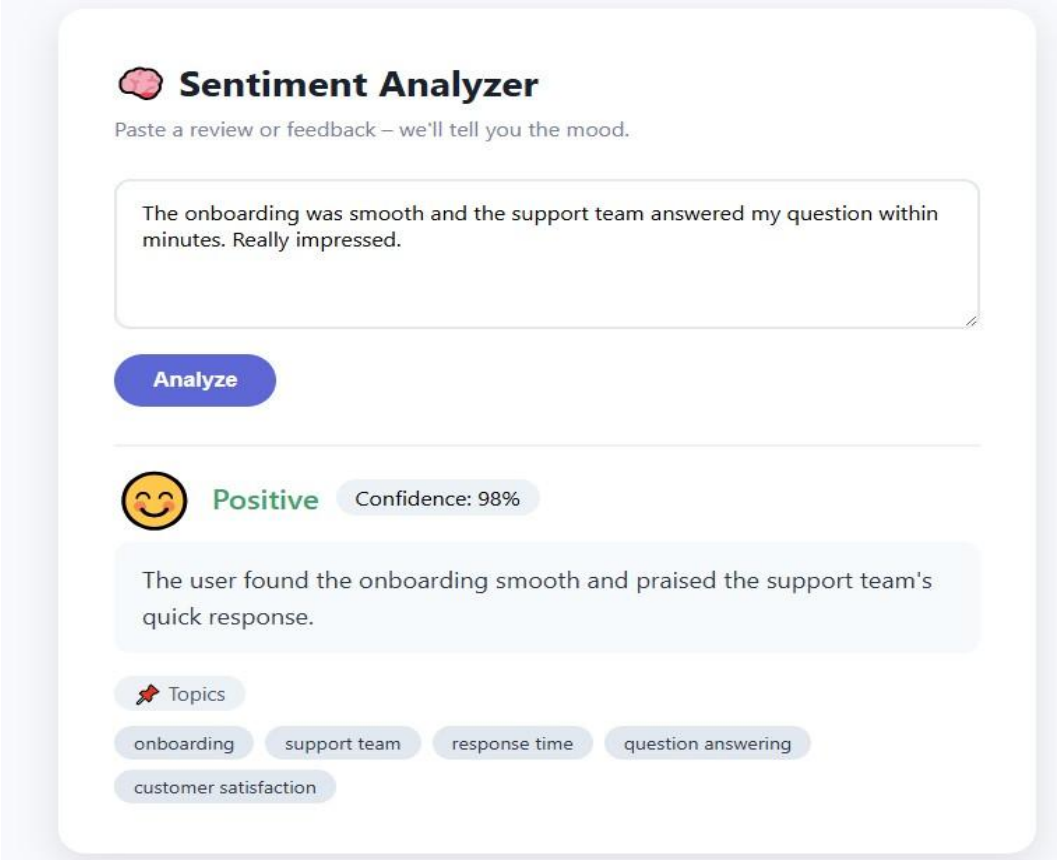
```
            form.addEventListener('submit', async function(e) {
                e.preventDefault();

                const text = textInput.value.trim();
                if (text.length < 3) {
                    showError('Please enter at least 3 characters.');
                    return;
                }

                // Reset UI
                hideError();
                resultDiv.classList.remove('visible');
                resultDiv.className = 'result';
                submitBtn.disabled = true;
                btnText.textContent = 'Analyzing...';
                loader.style.display = 'inline-block';

                try {
                    const response = await fetch('/api/sentiment/analyze', {
                        method: 'POST',
                        headers: {
                            'Content-Type': 'application/json',
                            'X-CSRF-TOKEN': document.querySelector('meta[name="csrf-token"]')?.content
|| '',
                        },
                        body: JSON.stringify({ text })
                    });

                    const data = await response.json();

                    if (!response.ok) {
                        throw new Error(data.message || 'Server error');
                    }

                    if (!data.success) {
                        throw new Error(data.message || 'Analysis failed');
                    }

                    displayResult(data.data);

                } catch (err) {
                    showError(err.message || 'Something went wrong. Please try again.');
                } finally {
                    submitBtn.disabled = false;
                    btnText.textContent = 'Analyze';
                    loader.style.display = 'none';
                }
            });

            function displayResult(data) {
                const { sentiment, confidence, topics, summary } = data;

                // Emoji
                emojiSpan.textContent = emojiMap[sentiment] || '😐';

                // Label
                sentimentLabel.textContent = sentiment.charAt(0).toUpperCase() + sentiment.slice(1);
```

```
                // Confidence (percentage, rounded)
                const confPercent = Math.round(confidence * 100);
                confidenceValue.textContent = confPercent;

                // Summary
                summaryText.textContent = summary || 'No summary provided.';

                // Topics
                topicsContainer.innerHTML = '';
                if (topics && topics.length) {
                    topics.forEach(topic => {
                        const tag = document.createElement('span');
                        tag.className = 'topic-tag';
                        tag.textContent = topic;
                        topicsContainer.appendChild(tag);
                    });
                } else {
                    const none = document.createElement('span');
                    none.className = 'topic-tag';
                    none.textContent = 'None detected';
                    topicsContainer.appendChild(none);
                }

                // Add sentiment class for color
                resultDiv.classList.add('sentiment-' + sentiment);
                resultDiv.classList.add('visible');
            }

            function showError(msg) {
                errorDiv.textContent = msg;
                errorDiv.classList.add('visible');
            }

            function hideError() {
                errorDiv.classList.remove('visible');
                errorDiv.textContent = '';
            }
        })();
    </script>
</body>
</html>
```

Example Results

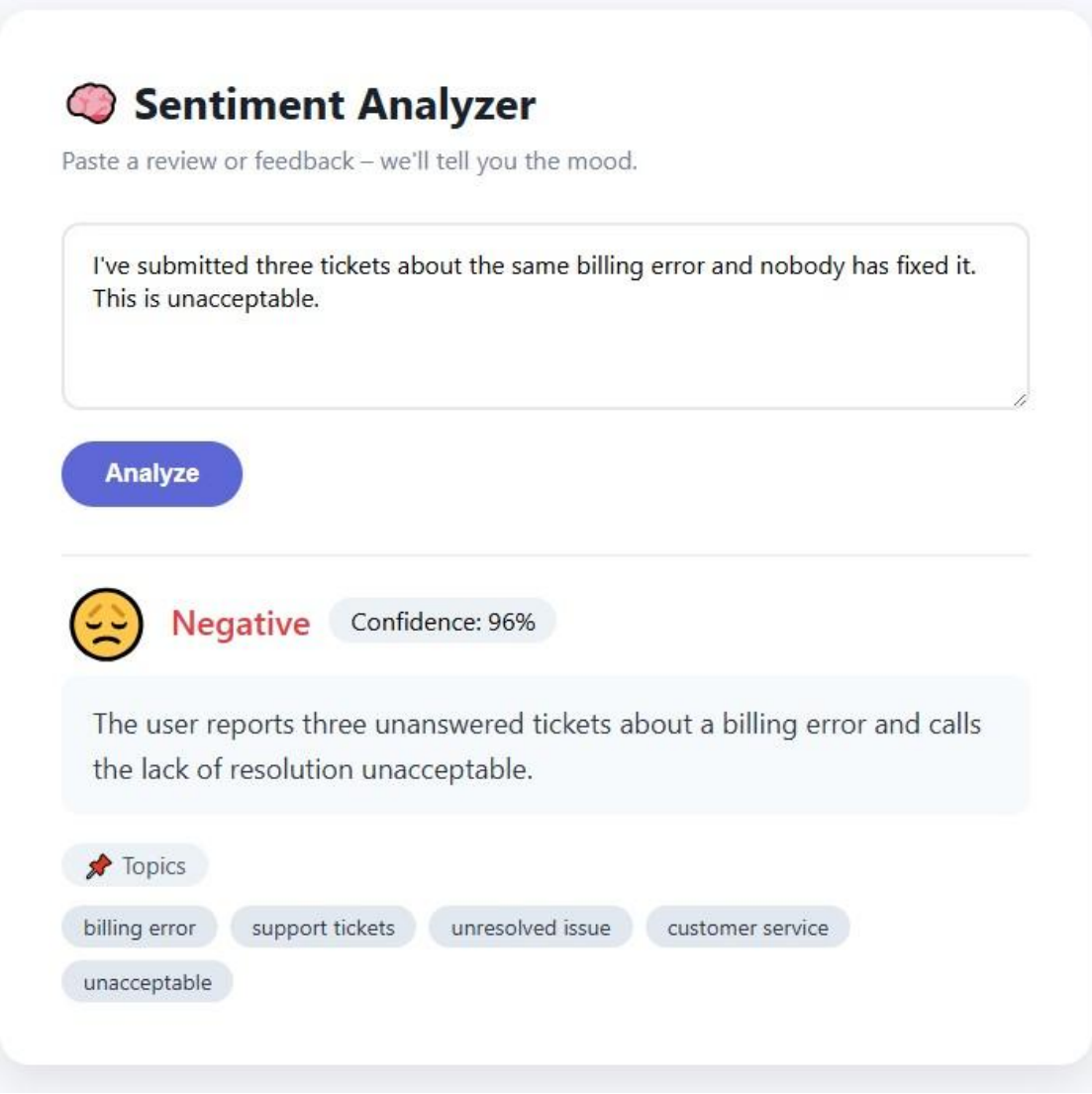
1. Clearly positive feedback

Example: "The onboarding was smooth and the support team answered my question within minutes. Really impressed."



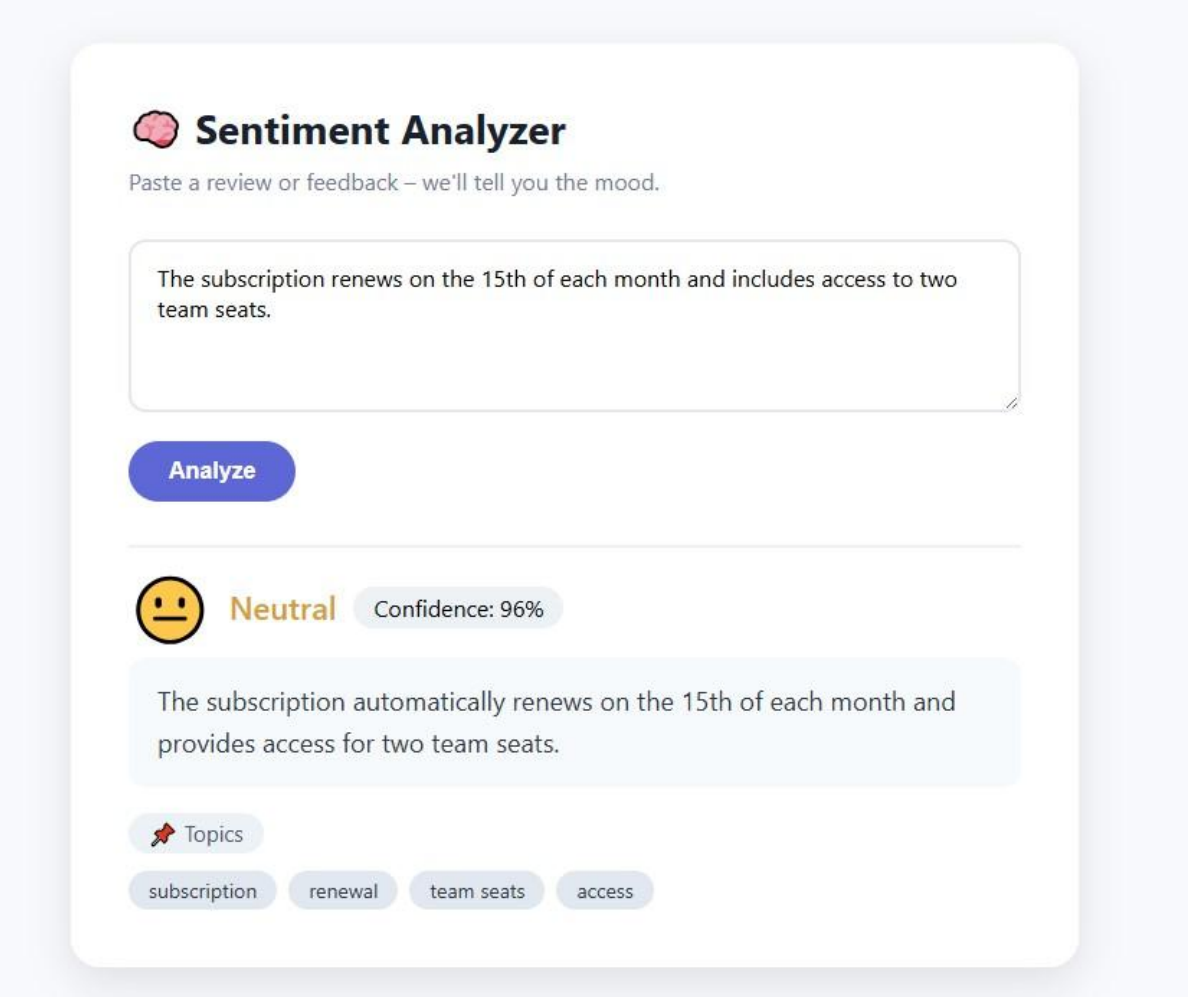
2. Clearly negative feedback

Example: "I've submitted three tickets about the same billing error and nobody has fixed it. This is unacceptable."



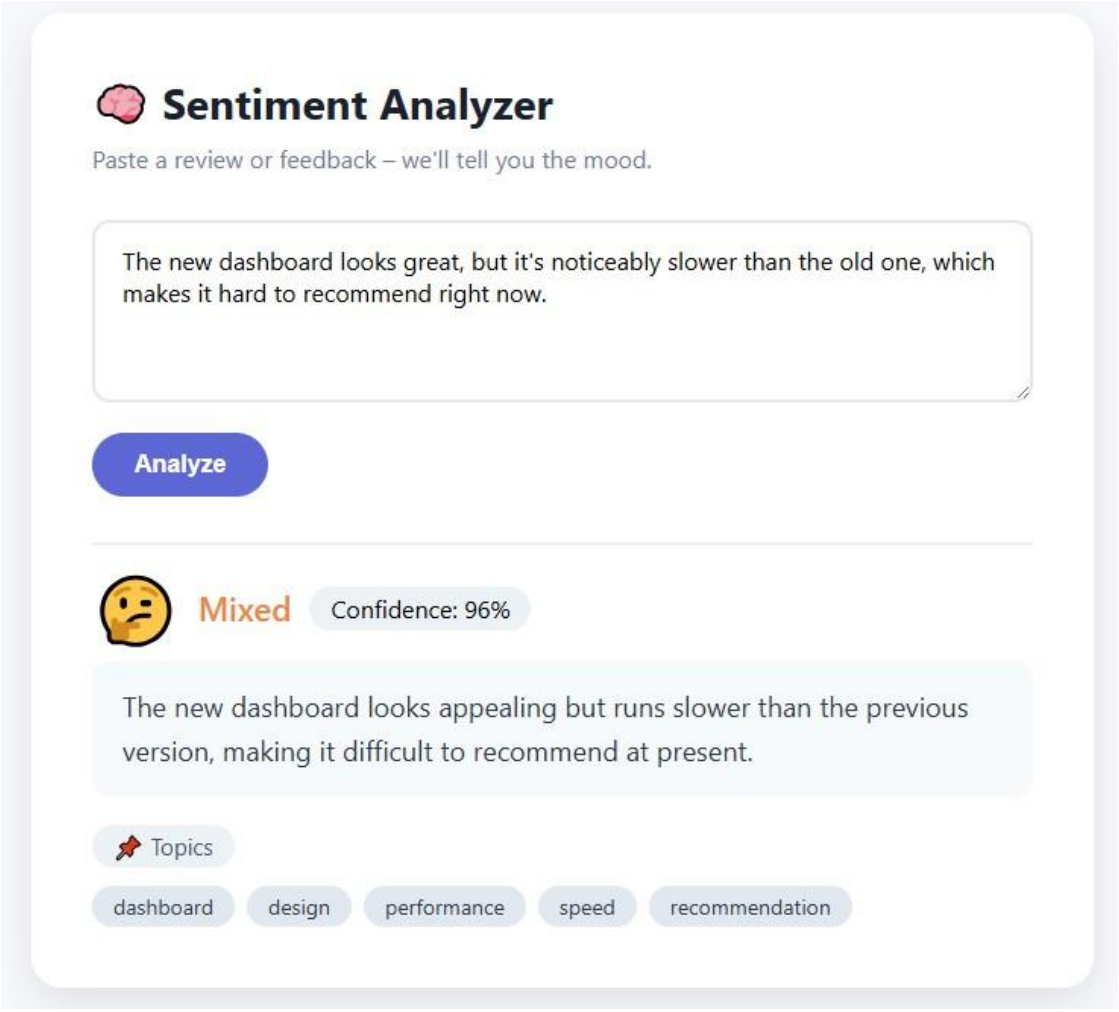
3. Neutral factual text

Example: "The subscription renews on the 15th of each month and includes access to two team seats."



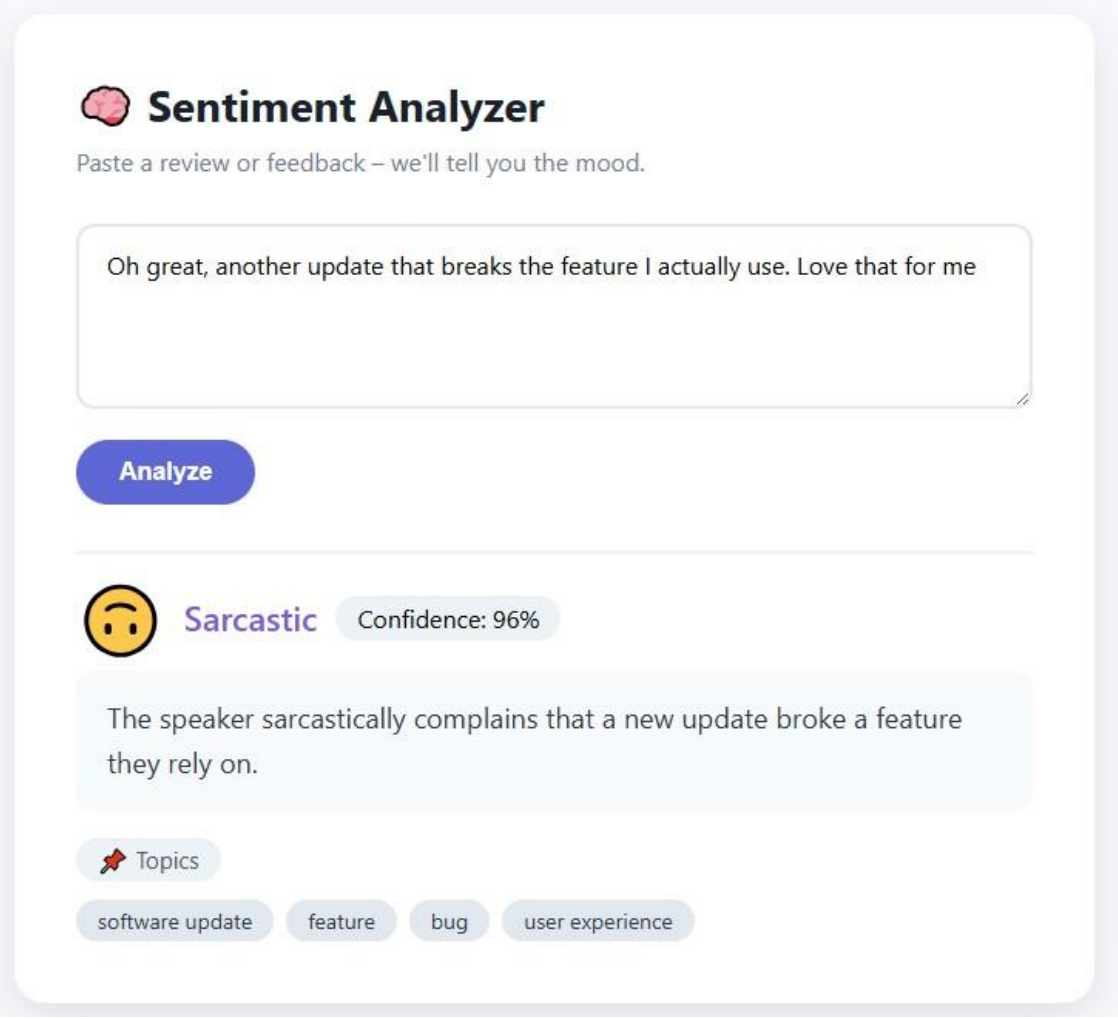
4. Mixed sentiment

Example: "The new dashboard looks great, but it's noticeably slower than the old one, which makes it hard to recommend right now."



5. Sarcasm

Example: "Oh great, another update that breaks the feature I actually use. Love that for me."



6. Emotionally ambiguous feedback

Example: "It's fine, I guess. Does what it says on the box, nothing more."