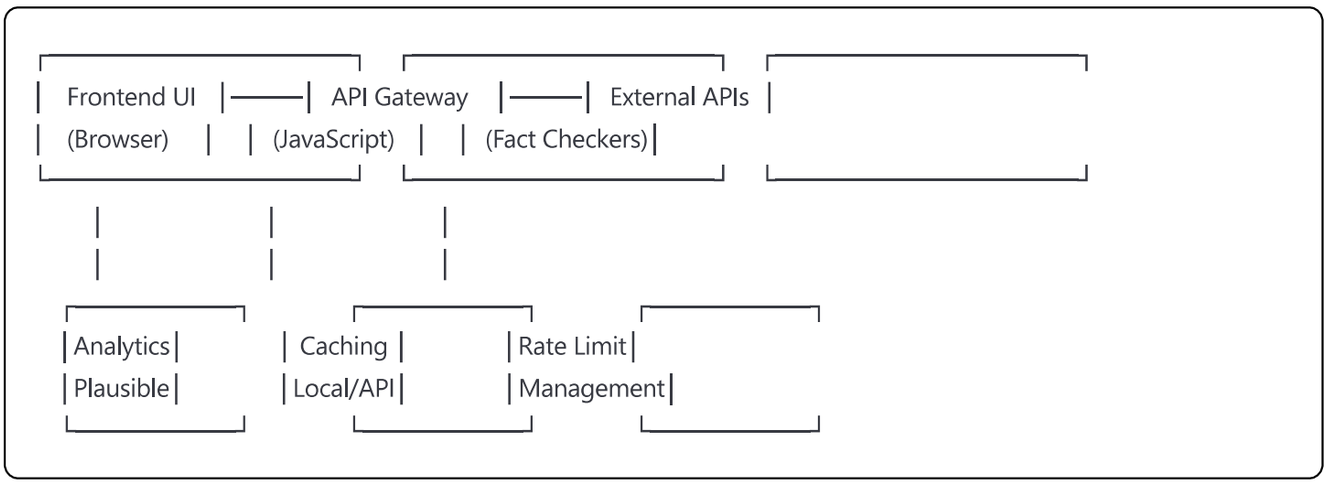
Technical Architecture Documentation

🏗️ System Architecture Overview

High-Level Architecture



Component Breakdown

1. Frontend Layer

- **Technology**: Vanilla HTML5, CSS3, ES6+ JavaScript
- **Architecture**: Single Page Application (SPA)
- **Responsiveness**: Mobile-first design with CSS Grid/Flexbox
- **State Management**: Local variables and DOM manipulation

2. API Integration Layer

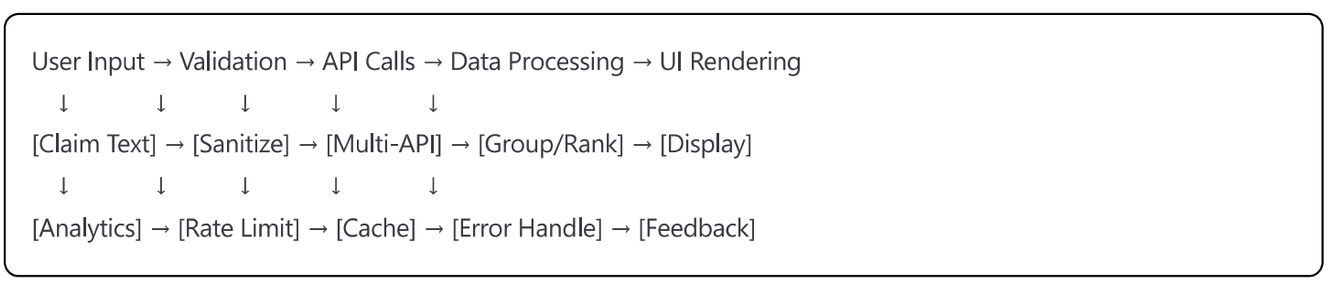
- **Primary**: Google Custom Search API
- **Secondary**: NewsAPI for broader coverage
- **Future**: AllSides API, Newstrition API
- **Error Handling**: Graceful degradation with fallback sources

3. Data Processing Layer

- **Source Grouping**: Publisher-based article organization
- **Content Filtering**: Fact-checker site prioritization
- **Result Ranking**: Relevance and credibility scoring

📊 Data Flow Architecture

User Interaction Flow



API Call Strategy

javascript

```
// Sequential API calling with fallback
async function performFactCheck(claim) {
    const results = [];

    try {
        // Primary: Google Custom Search
        const googleResults = await searchGoogleFactCheckers(claim);
        results.push(...googleResults);
    } catch (error) {
        console.warn('Google API failed:', error);
    }

    try {
        // Secondary: NewsAPI
        const newsResults = await searchNewsAPI(claim);
        results.push(...newsResults);
    } catch (error) {
        console.warn('NewsAPI failed:', error);
    }

    // Process and deduplicate
    return processResults(results);
}
```

🔧 Core Components

1. Search Engine (`searchManager.js`)

javascript

```
class SearchManager {
    constructor() {
        this.apiKeys = CONFIG;
        this.rateLimit = new RateLimit(60, 60000); // 60/minute
        this.cache = new Map();
    }

    async search(query) {
        // Rate limiting
        if (!this.rateLimit.check()) {
            throw new Error('Rate limit exceeded');
        }

        // Cache check
        const cacheKey = this.hashQuery(query);
        if (this.cache.has(cacheKey)) {
            return this.cache.get(cacheKey);
        }

        // Execute search
        const results = await this.performSearch(query);
        this.cache.set(cacheKey, results);
        return results;
    }
}
```

2. Result Processor (`resultProcessor.js`)

javascript

```
class ResultProcessor {
    static groupByPublisher(articles) {
        const grouped = new Map();

        for (const article of articles) {
            const publisher = this.extractPublisher(article.link);

            if (!grouped.has(publisher)) {
                grouped.set(publisher, []);
            }
            grouped.get(publisher).push(article);
        }

        return grouped;
    }

    static rankSources(grouped) {
        const reliability = {
            'factcheck.org': 10,
            'snopes.com': 9,
            'politifact.com': 9,
            'factcheck.afp.com': 8,
            'fullfact.org': 8
        };

        return new Map([...grouped.entries()].sort((a, b) => {
            const scoreA = reliability[a[0]] || 5;
            const scoreB = reliability[b[0]] || 5;
            return scoreB - scoreA;
        }));
    }
}
```

3. UI Manager (`uiManager.js`)

javascript

```
class UIManager {
    constructor() {
        this.resultsContainer = document.getElementById('results');
        this.loadingState = false;
        this.currentResults = null;
    }

    displayResults(groupedResults) {
        this.resultsContainer.innerHTML = '';

        for (const [publisher, articles] of groupedResults) {
            const sourceElement = this.createSourceElement(publisher, articles);
            this.resultsContainer.appendChild(sourceElement);
        }

        this.updateStats(groupedResults);
        this.attachEventListeners();
    }

    createSourceElement(publisher, articles) {
        const [primary, ...additional] = articles;

        return `
            <div class="source-group">
                <div class="primary-result">${this.formatArticle(primary)}</div>
                ${additional.length > 0 ? this.createExpandableSection(additional) : ''}
            </div>
        `;
    }
}
```

🔌 API Integration Details

Google Custom Search API

```
javascript
```

```javascript
const GOOGLE_CONFIG = {
    baseUrl: 'https://www.googleapis.com/customsearch/v1',
    parameters: {
        key: CONFIG.GOOGLE_API_KEY,
        cx: CONFIG.GOOGLE_SEARCH_ENGINE_ID,
        q: '', // User query
        siteSearch: 'factcheck.org OR snopes.com OR politifact.com',
        num: 10,
        fields: 'items(title,link,snippet,displayLink)'
    }
};

async function searchGoogleFactCheckers(query) {
    const url = new URL(GOOGLE_CONFIG.baseUrl);
    const params = { ...GOOGLE_CONFIG.parameters, q: query };

    Object.entries(params).forEach(([key, value]) => {
        url.searchParams.append(key, value);
    });

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

    const data = await response.json();
    return data.items || [];
}
```

NewsAPI Integration

javascript

```
const NEWS_CONFIG = {
    baseUrl: 'https://newsapi.org/v2/everything',
    parameters: {
        apiKey: CONFIG.NEWS_API_KEY,
        q: '', // User query
        domains: 'factcheck.org,snopes.com,politifact.com,factcheck.afp.com',
        sortBy: 'relevancy',
        pageSize: 20,
        language: 'en'
    }
};

async function searchNewsAPI(query) {
    const url = new URL(NEWS_CONFIG.baseUrl);
    const params = { ...NEWS_CONFIG.parameters, q: query };

    Object.entries(params).forEach(([key, value]) => {
        url.searchParams.append(key, value);
    });

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

    const data = await response.json();
    return data.articles || [];
}
```

🎨 Frontend Architecture

CSS Architecture

```
css
```

```css
/* Component-based CSS structure */

/* Base styles */
:root {
    --primary-color: #2c3e50;
    --secondary-color: #3498db;
    --success-color: #27ae60;
    --warning-color: #f39c12;
    --error-color: #e74c3c;
    --font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', sans-serif;
}

/* Layout components */
.container { /* Main layout */ }
.header { /* Top navigation */ }
.search-section { /* Input area */ }
.results-section { /* Output area */ }
.footer { /* Bottom info */ }

/* Interactive components */
.source-group { /* Publisher grouping */ }
.expandable-section { /* Collapsible content */ }
.loading-spinner { /* Progress indicator */ }
.error-message { /* Error display */ }

/* Responsive breakpoints */
@media (max-width: 768px) { /* Mobile styles */ }
@media (min-width: 769px) and (max-width: 1024px) { /* Tablet styles */ }
@media (min-width: 1025px) { /* Desktop styles */ }
```

JavaScript Module Structure

javascript

```javascript
// Main application module
const FactChecker = {
    // Core modules
    search: new SearchManager(),
    processor: new ResultProcessor(),
    ui: new UIManager(),
    analytics: new AnalyticsManager(),

    // Application lifecycle
    async init() {
        this.bindEventListeners();
        this.loadConfiguration();
        this.analytics.initialize();
    },

    async performFactCheck(claim) {
        try {
            this.ui.showLoading();

            const results = await this.search.query(claim);
            const processed = this.processor.groupAndRank(results);

            this.ui.displayResults(processed);
            this.analytics.trackSearch(claim, results.length);

        } catch (error) {
            this.ui.showError(error.message);
            this.analytics.trackError(error);
        } finally {
            this.ui.hideLoading();
        }
    }
};
```

📊 Performance Optimization

Caching Strategy

javascript

```
class CacheManager {
    constructor() {
        this.cache = new Map();
        this.maxSize = 100;
        this.ttl = 300000; // 5 minutes
    }

    set(key, value) {
        // LRU eviction
        if (this.cache.size >= this.maxSize) {
            const firstKey = this.cache.keys().next().value;
            this.cache.delete(firstKey);
        }

        this.cache.set(key, {
            value,
            timestamp: Date.now()
        });
    }

    get(key) {
        const item = this.cache.get(key);
        if (!item) return null;

        // Check TTL
        if (Date.now() - item.timestamp > this.ttl) {
            this.cache.delete(key);
            return null;
        }

        return item.value;
    }
}
```

Rate Limiting

javascript

```
class RateLimit {
    constructor(maxRequests, windowMs) {
        this.maxRequests = maxRequests;
        this.windowMs = windowMs;
        this.requests = [];
    }

    check() {
        const now = Date.now();
        const windowStart = now - this.windowMs;

        // Remove expired requests
        this.requests = this.requests.filter(time => time > windowStart);

        if (this.requests.length >= this.maxRequests) {
            return false;
        }

        this.requests.push(now);
        return true;
    }
}
```

🔒 Security Considerations

Input Sanitization

```
javascript

function sanitizeInput(input) {
    // Remove dangerous characters
    const sanitized = input
        .replace(/<script\b[^<]*(?:(?!<\/script>)<[^<]*)*<\/script>/gi, '')
        .replace(/[<>]/g, '')
        .trim();

    // Length limit
    if (sanitized.length > 500) {
        throw new Error('Query too long');
    }

    return sanitized;
}
```

API Key Protection

- Environment variables for production
- Example configuration files in repository
- .gitignore for sensitive files
- Client-side key rotation capability

📈 Analytics Architecture

Event Tracking

```javascript
class AnalyticsManager {
    constructor() {
        this.plausible = window.plausible;
        this.enabled = CONFIG.ENABLE_ANALYTICS;
    }

    trackSearch(query, resultCount) {
        if (!this.enabled) return;

        this.plausible('Fact Check Search', {
            props: {
                resultCount,
                queryLength: query.length,
                hasResults: resultCount > 0
            }
        });
    }

    trackSourceExpansion(publisher) {
        if (!this.enabled) return;

        this.plausible('Source Expanded', {
            props: { publisher }
        });
    }
}
```

🚀 Deployment Architecture

Static Hosting Requirements

- **Server**: Any static file server (Apache, Nginx, GitHub Pages)
- **SSL**: HTTPS required for secure API calls
- **CDN**: Optional for global performance
- **Monitoring**: Uptime monitoring recommended

Environment Configuration

```javascript
// Production environment
const PRODUCTION_CONFIG = {
    API_ENDPOINTS: {
        google: 'https://googleapis.com/customsearch/v1',
        news: 'https://newsapi.org/v2'
    },
    RATE_LIMITS: {
        search: { requests: 100, window: 3600000 }, // 100/hour
        api: { requests: 1000, window: 86400000 }   // 1000/day
    },
    CACHE_TTL: 300000, // 5 minutes
    DEBUG_MODE: false
};
```

This architecture provides a scalable, maintainable foundation for the fact-checking tool with clear separation of concerns and robust error handling! 🏗️