

GenAI-Scout Lite - Architecture & C4 Documentation

Table of Contents

- 1. [System Overview](#system-overview)
- 2. [C4 Model Documentation](#c4-model-documentation)
- 3. [Technical Architecture](#technical-architecture)
- 4. [Data Flow Architecture](#data-flow-architecture)
- 5. [Security Architecture](#security-architecture)
- 6. [Deployment Architecture](#deployment-architecture)
- 7. [API Architecture](#api-architecture)
- 8. [Component Details](#component-details)

System Overview

GenAI-Scout Lite is a simplified technology scouting assistant designed to provide AI-powered technology intelligence for small to mid-sized enterprises and innovation teams. The system leverages generative AI to automate reading, summarizing, and profiling technology insights.

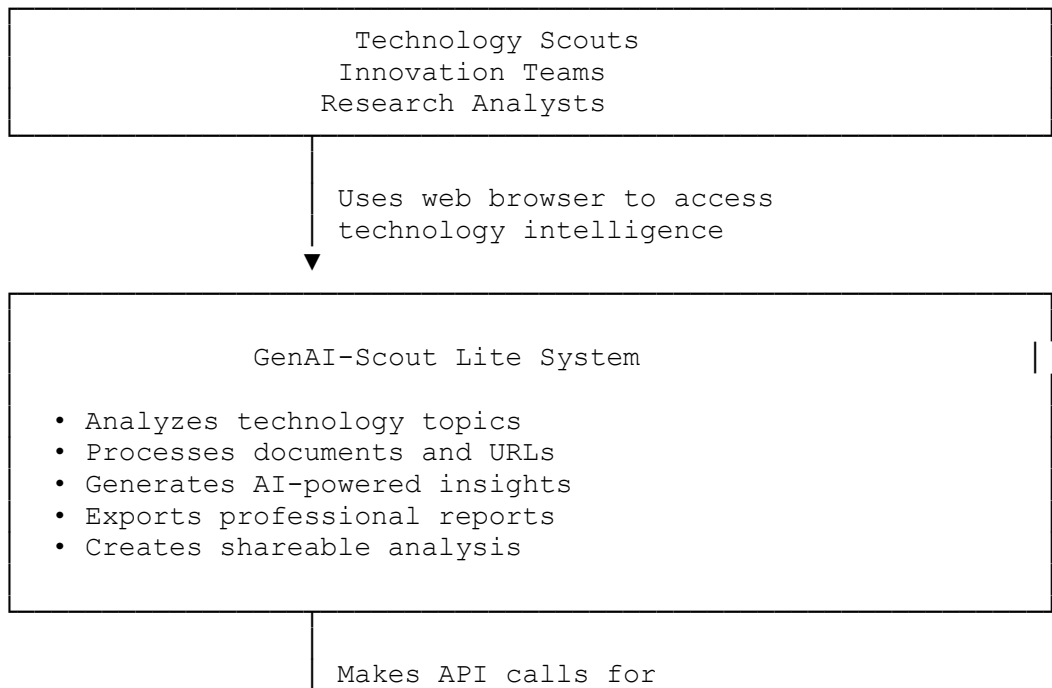
Key Principles

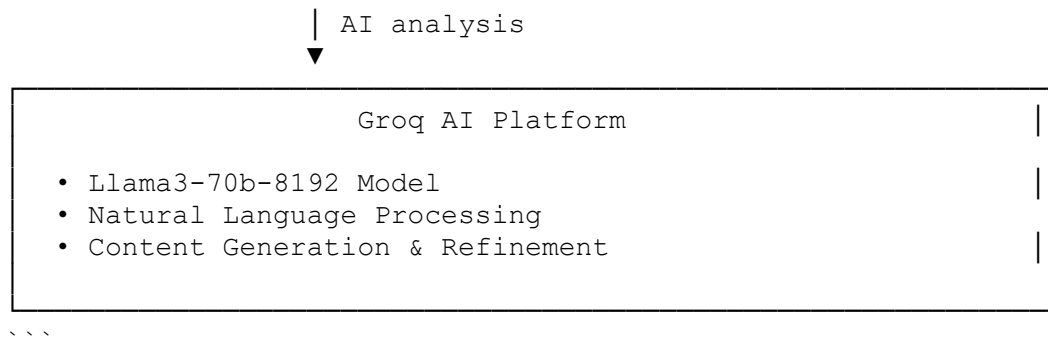
- **No-Login Access**: Immediate functionality without user registration
- **AI-First Approach**: Groq API integration for fast, reliable analysis
- **Multi-Input Support**: Topic text, document upload, and URL analysis
- **Export Flexibility**: PDF, Markdown, and shareable links
- **Professional UI**: Advanced aesthetics suitable for business presentations

C4 Model Documentation

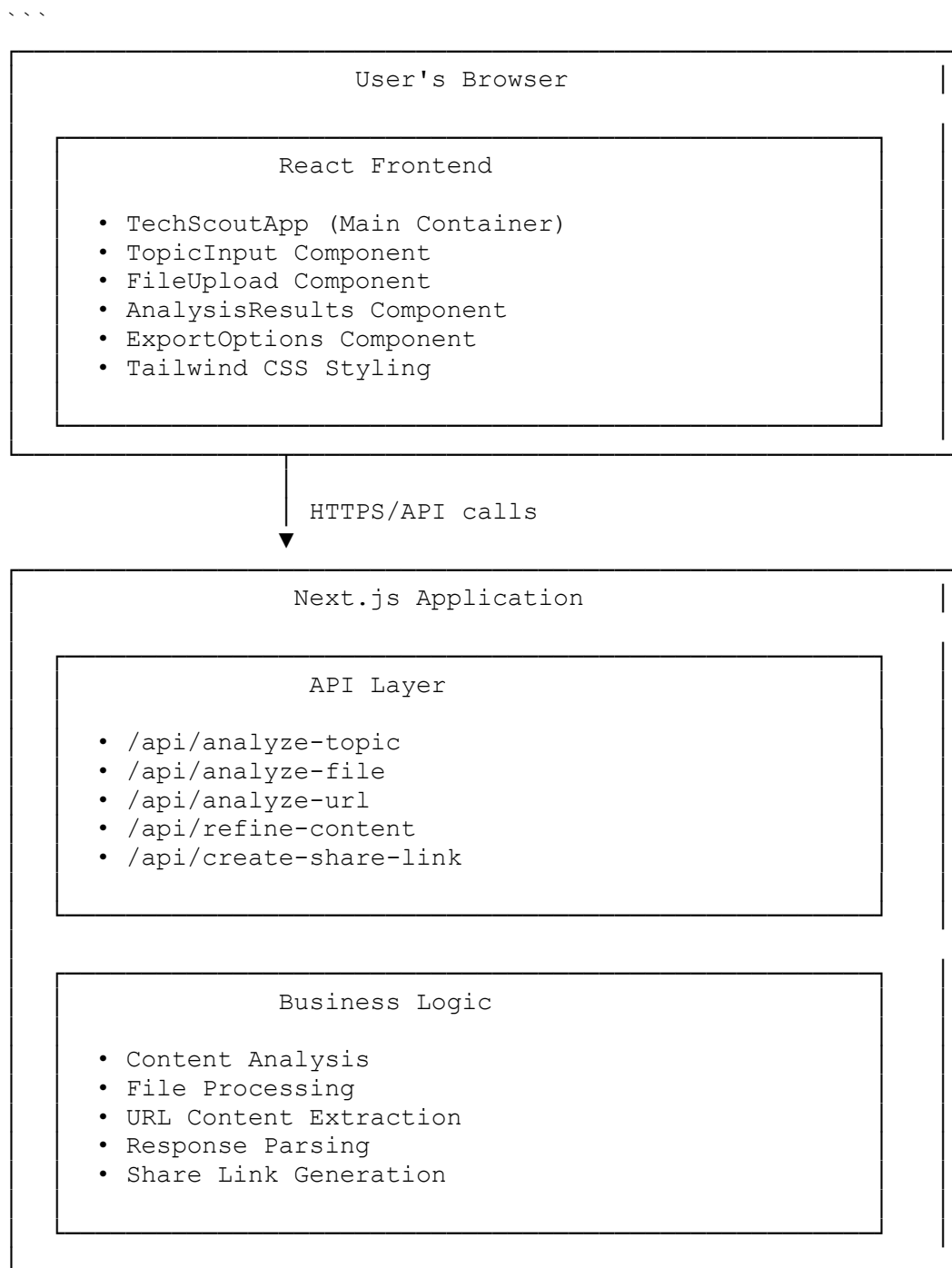
Level 1: System Context Diagram

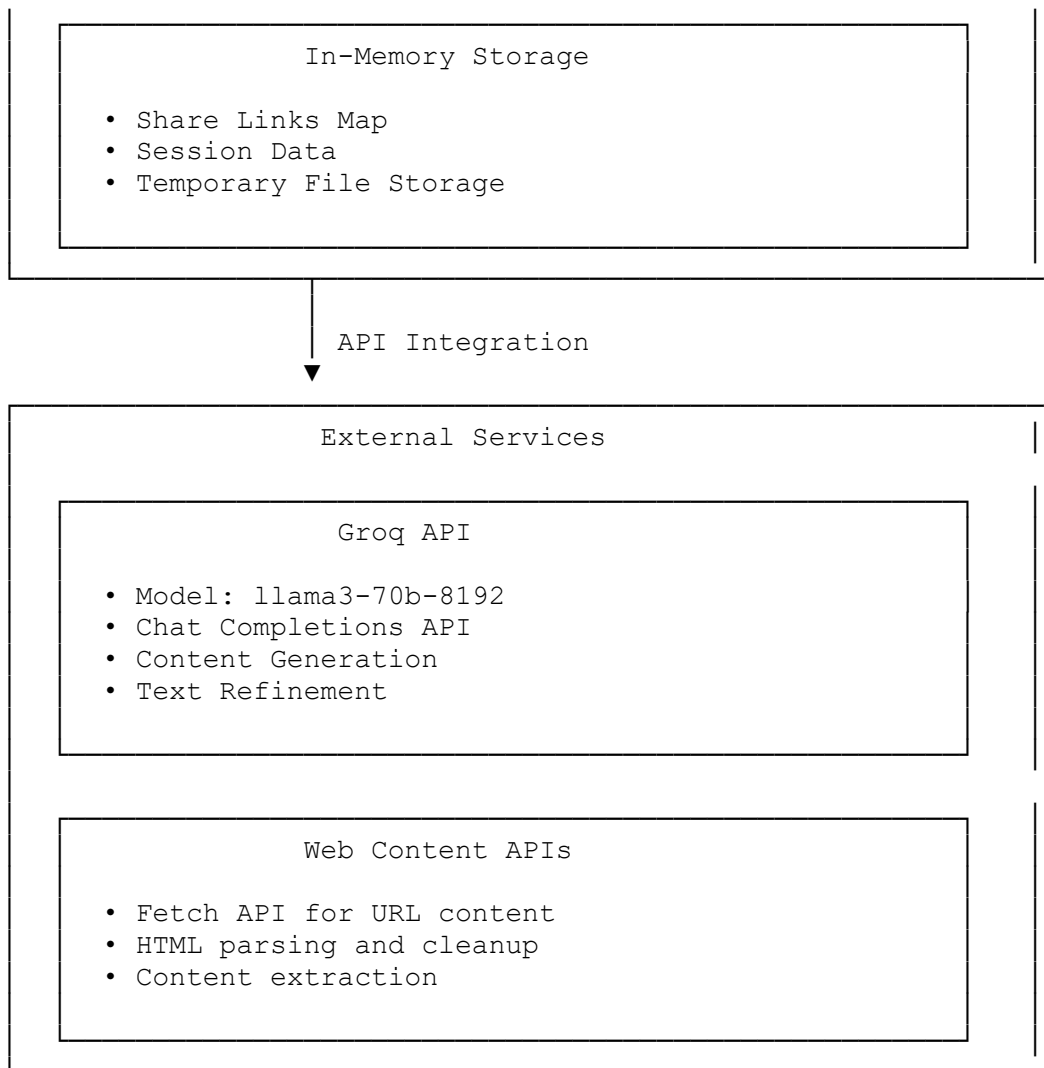
...





Level 2: Container Diagram

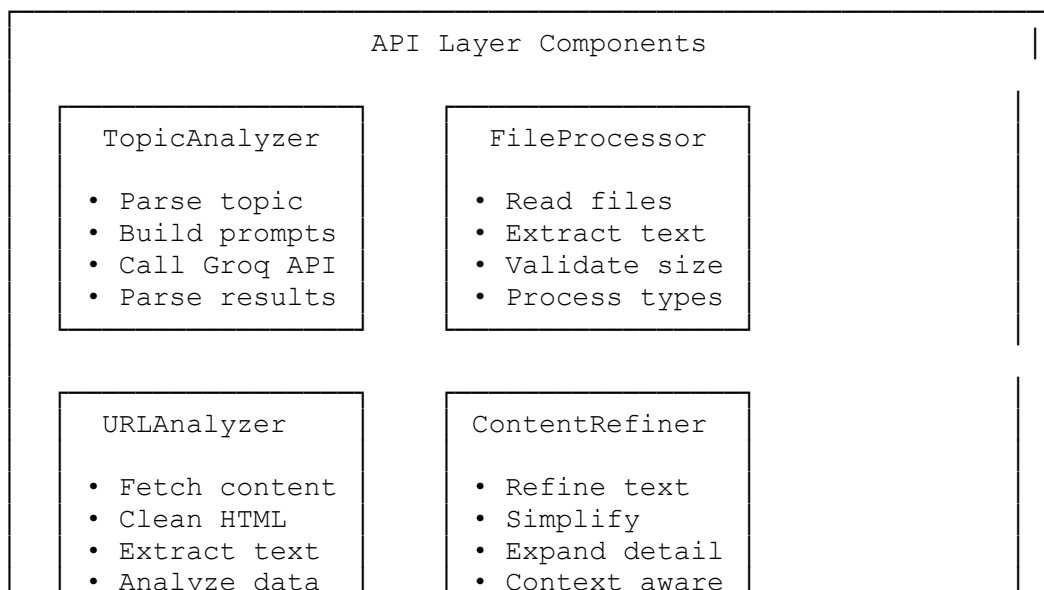


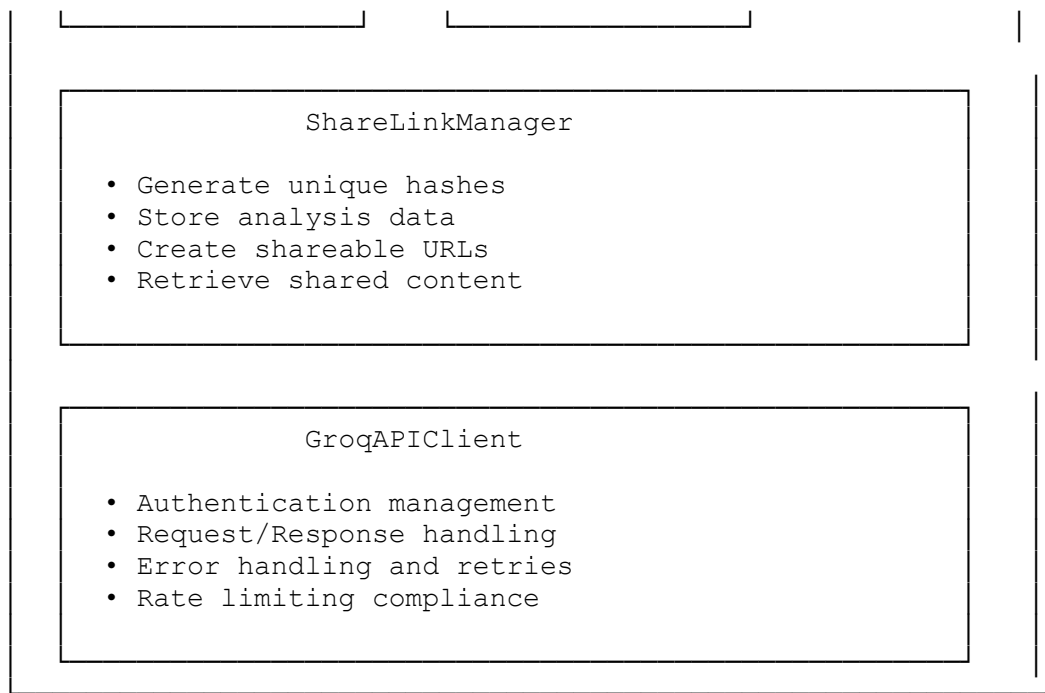


...

Level 3: Component Diagram - API Layer

...





— — —

Level 4: Code Structure

— — —

```

src/
├── app/
│   ├── api/
│   │   ├── analyze-topic/
│   │   │   └── route.ts
│   │   ├── analyze-file/
│   │   │   └── route.ts
│   │   ├── analyze-url/
│   │   │   └── route.ts
│   │   ├── refine-content/
│   │   │   └── route.ts
│   │   └── create-share-link/
│   │       └── route.ts
│   ├── share/
│   │   └── [shareId]/
│   │       └── page.tsx
│   ├── layout.tsx
│   └── page.tsx
├── components/
│   ├── TechScoutApp.tsx
│   ├── TopicInput.tsx
│   ├── FileUpload.tsx
│   ├── AnalysisResults.tsx
│   └── ExportOptions.tsx
└── styles/
    └── globals.css

```

API Routes

Topic analysis endpoint

File analysis endpoint

URL analysis endpoint

Content refinement endpoint

Share link generation

Shared analysis viewer

Root layout

Home page

Main application container

Topic input component

File upload component

Results display component

Export functionality

Global styles

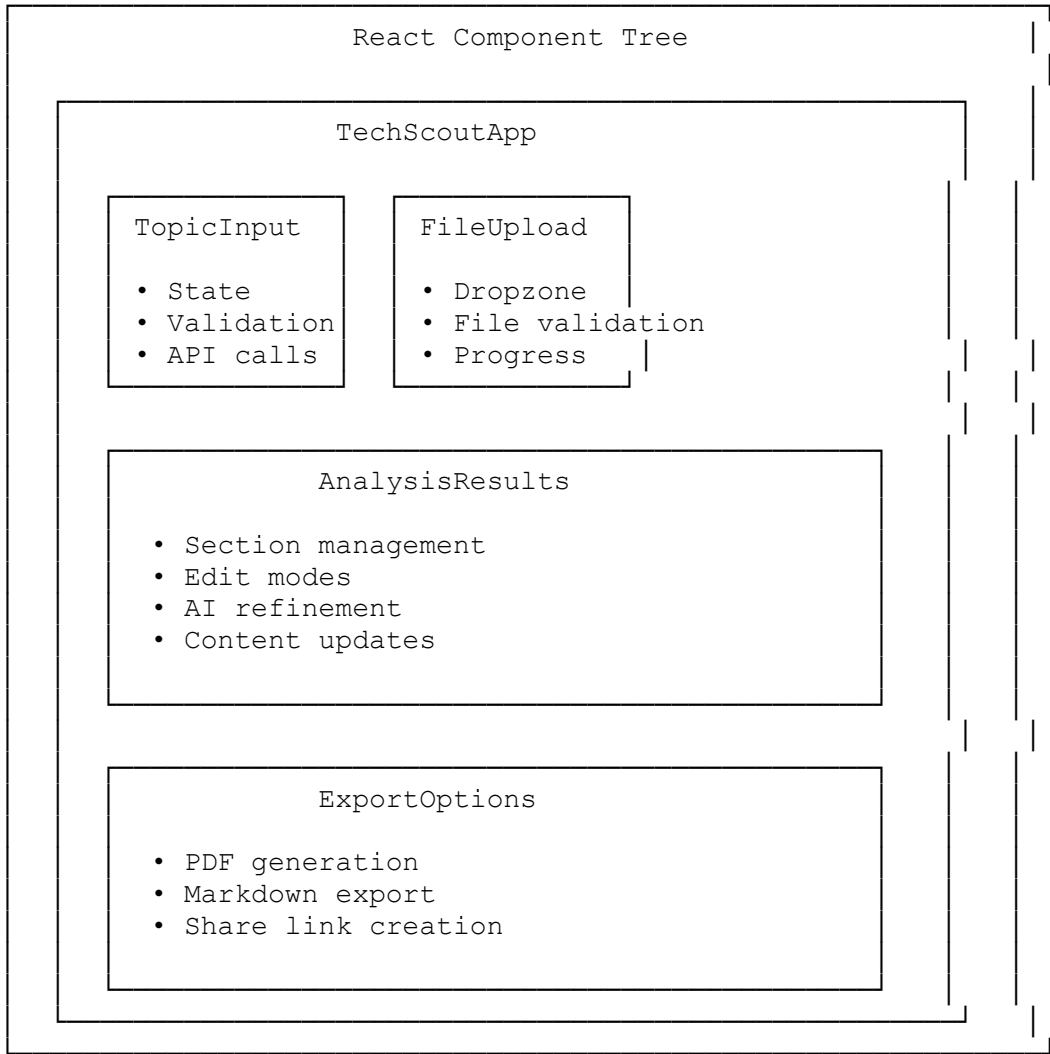
— — —

— — —

Technical Architecture

Frontend Architecture

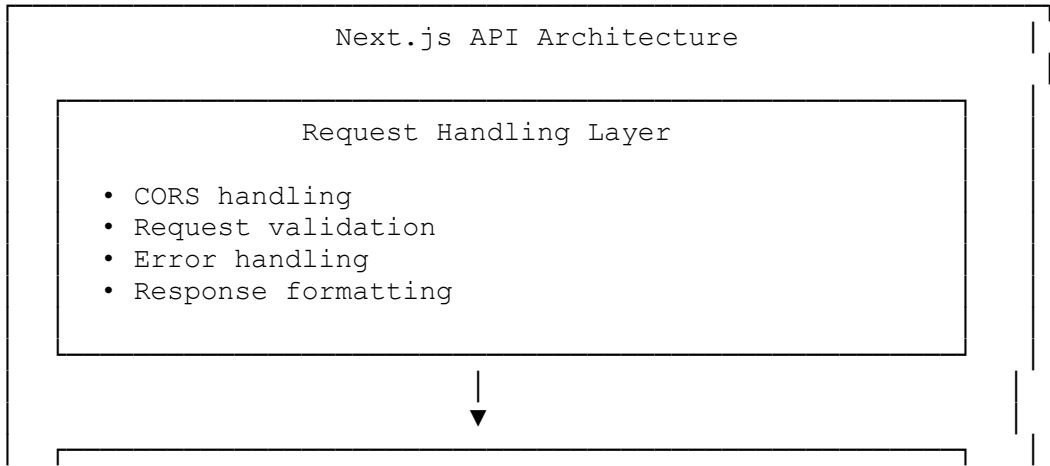
...

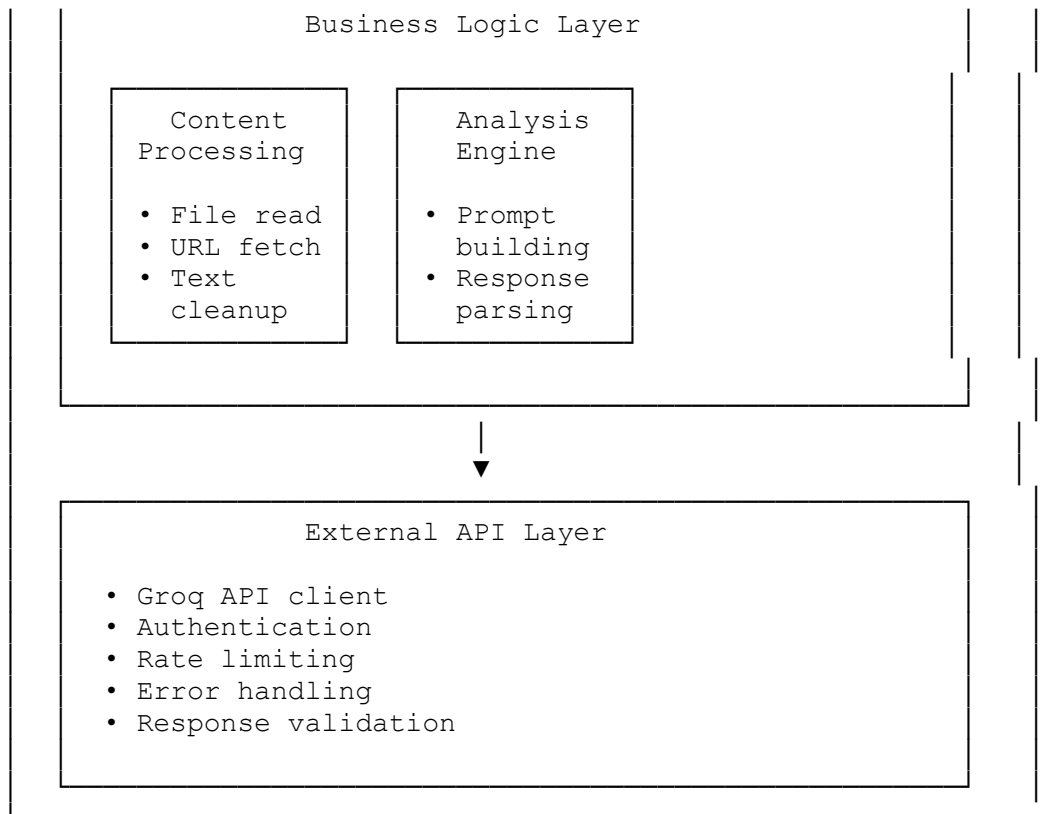


...

Backend Architecture

...





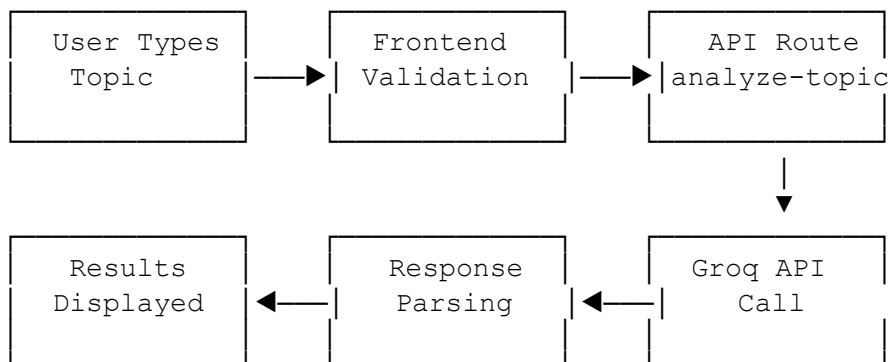
...

Data Flow Architecture

Topic Analysis Flow

...

User Input → Validation → Prompt Building → Groq API → Response Parsing → UI Update

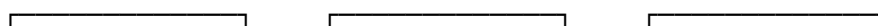


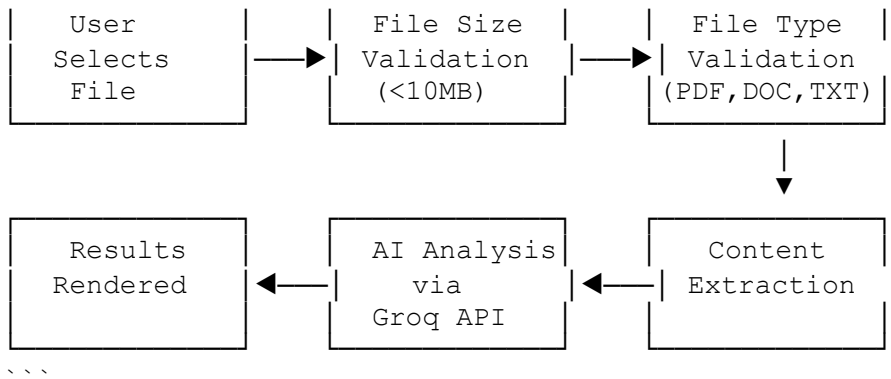
...

File Upload Flow

...

File Selection → Validation → Content Extraction → Analysis → Results

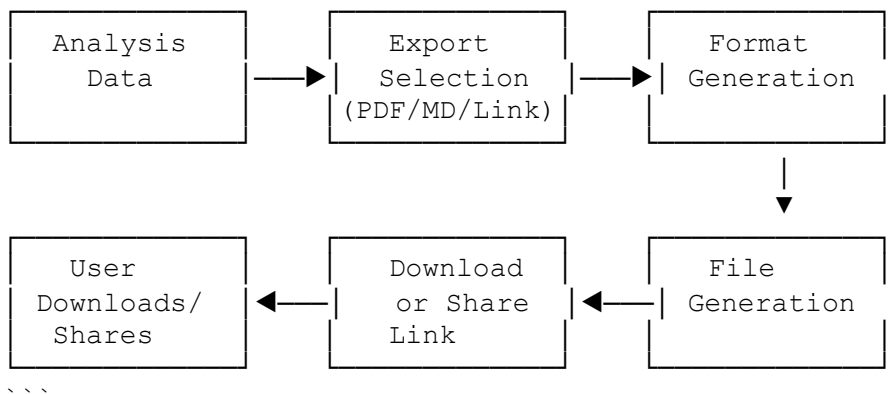




Export Flow

...

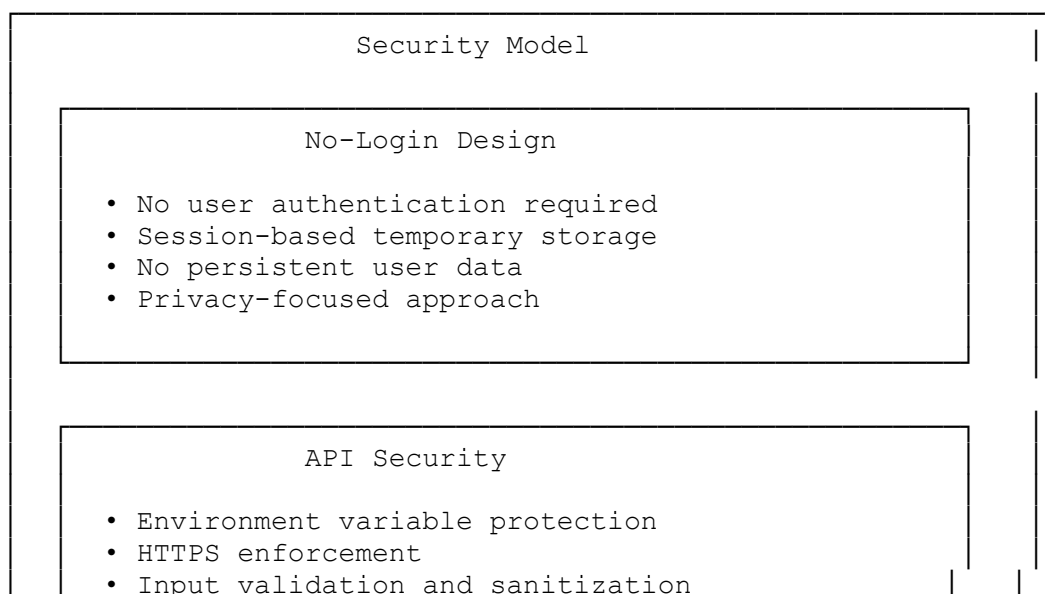
Analysis Data → Format Selection → Content Generation → Download/Share

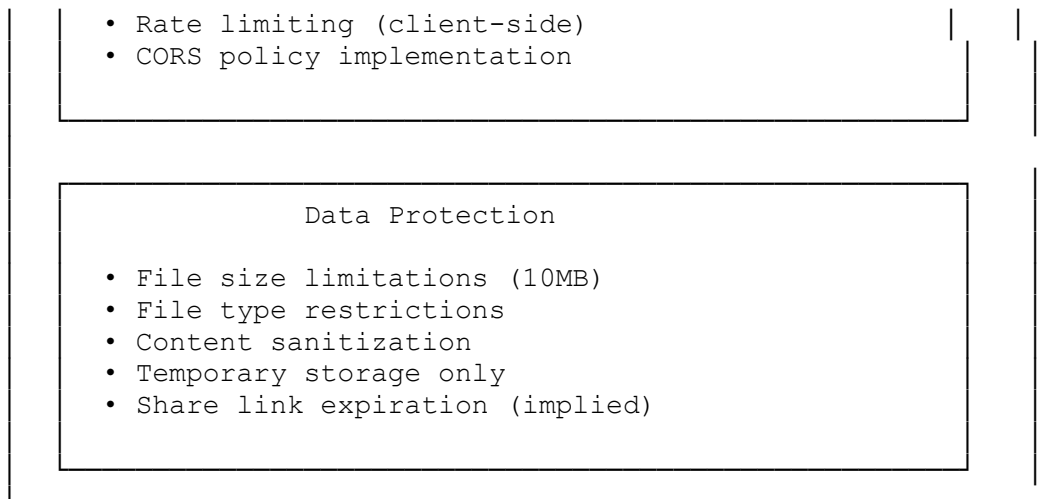


Security Architecture

Authentication & Authorization

...



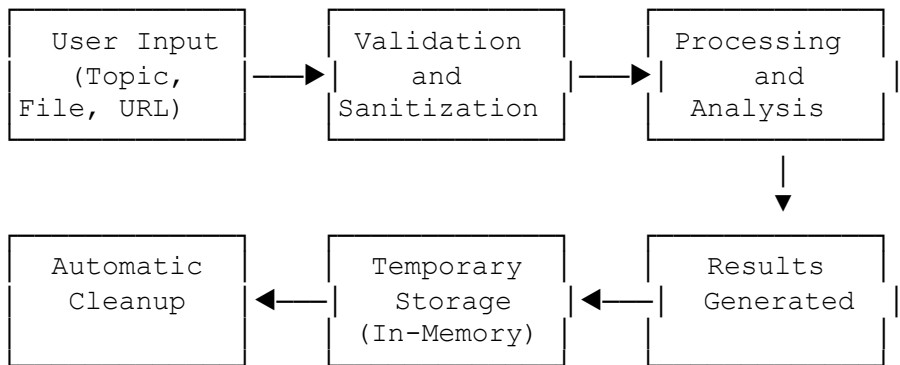


...

Data Privacy Flow

...

User Data → Validation → Processing → Analysis → Temporary Storage → Cleanup

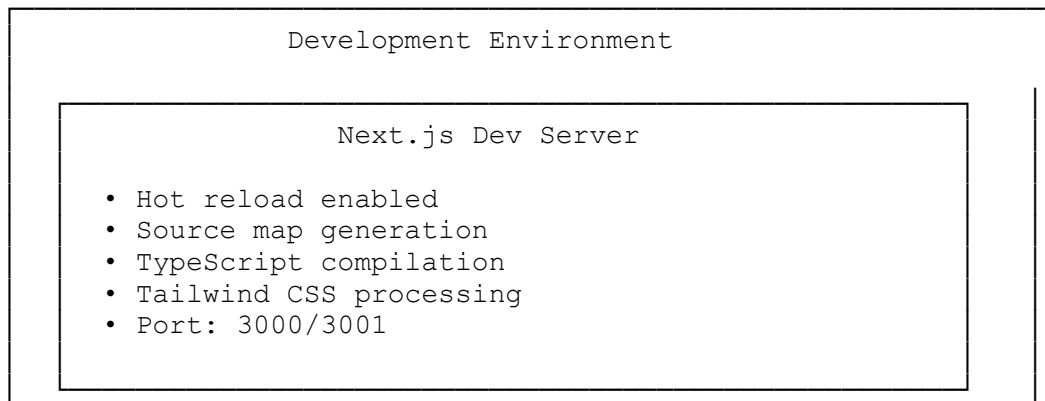


...

Deployment Architecture

Local Development

...



Environment Variables

- GROQ_API_KEY (from .env.local)
- NEXT_PUBLIC_APP_NAME

...

Production Deployment (Recommended: Vercel)

...

Vercel Deployment

Edge Functions

- API routes deployed as serverless functions
- Automatic scaling
- Global edge network
- Cold start optimization

Static Assets

- React components (SSG/SSR)
- CSS/JavaScript bundles
- Image optimization
- CDN distribution

Environment Security

- Encrypted environment variables
- HTTPS enforcement
- DDoS protection
- Automatic certificate management

...

API Architecture

RESTful API Design

...

API Endpoints

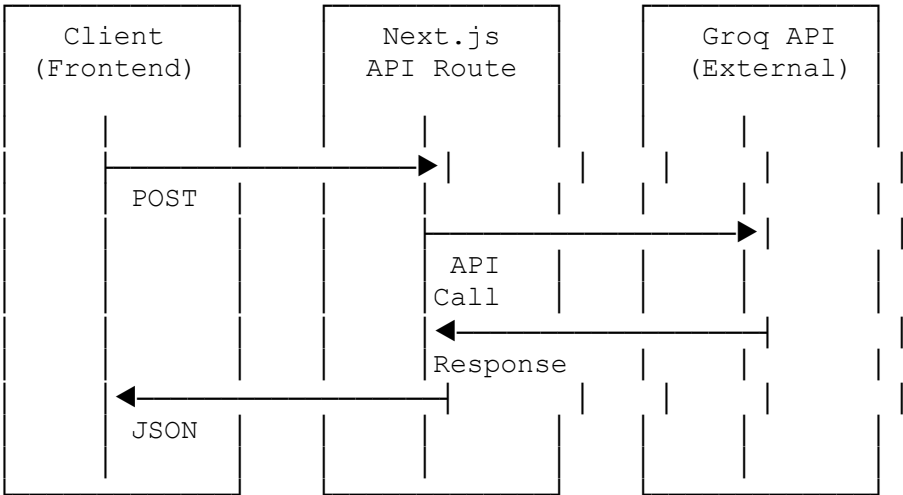
POST /api/analyze-topic	
Request: { topic: string }	
Response: { topic, summary, marketTrends, keyPlayers, useCases, challenges }	
Purpose: Analyze technology topics	
POST /api/analyze-file	
Request: FormData with file	
Response: { topic, summary, marketTrends, keyPlayers, useCases, challenges }	
Purpose: Extract insights from documents	
POST /api/analyze-url	
Request: { url: string }	
Response: { topic, summary, marketTrends, keyPlayers, useCases, challenges }	
Purpose: Analyze web page content	
POST /api/refine-content	
Request: { content, action, context }	
Response: { refinedContent }	
Purpose: AI-assisted content improvement	
POST /api/create-share-link	
Request: AnalysisData object	
Response: { shareableUrl, shareId }	
Purpose: Generate shareable analysis links	
GET /api/create-share-link/[shareId]	
Request: ShareId in URL	
Response: AnalysisData object	
Purpose: Retrieve shared analysis	

...

API Request/Response Flow

...

Client Request → Validation → Business Logic → External API → Response



...

Component Details

Frontend Components

TechScoutApp (Main Container)

```typescript

interface TechScoutApp {

state: {

currentStep: 'input' | 'analysis' | 'results';

analysisData: AnalysisData | null;

isLoading: boolean;

}

methods: {

handleAnalysisComplete: (data: AnalysisData) => void;

handleNewAnalysis: () => void;

}

responsibilities: [

'Manage application state',

'Route between different views',

'Handle step transitions',

'Provide global loading state'

]

}

```

TopicInput Component

```typescript

interface TopicInput {

props: {

onAnalysisComplete: (data: AnalysisData) => void;

}

state: {

topic: string;

isLoading: boolean;

}

responsibilities: [

'Capture user topic input',

'Validate input data',

'Call analysis API',

'Handle loading states',

'Provide example topics'

]

}

```

FileUpload Component

```typescript

interface FileUpload {

props: {

onAnalysisComplete: (data: AnalysisData) => void;

}

```

state: {
 uploadedFile: File | null;
 urlInput: string;
 isLoading: boolean;
}

responsibilities: [
 'Handle file drag & drop',
 'Validate file types and sizes',
 'Process URL inputs',
 'Upload files to API',
 'Manage upload progress'
]
}
```

#### AnalysisResults Component
```typescript
interface AnalysisResults {
 props: {
 data: AnalysisData;
 onDataUpdate: (data: AnalysisData) => void;
 }

 state: {
 editingSection: string | null;
 editingContent: string;
 isRefining: string | null;
 }

 responsibilities: [
 'Display analysis sections',
 'Handle inline editing',
 'Trigger content refinement',
 'Manage edit/save states',
 'Update parent with changes'
]
}
```

#### ExportOptions Component
```typescript
interface ExportOptions {
 props: {
 data: AnalysisData;
 }

 state: {
 isExporting: string | null;
 shareableLink: string | null;
 }

 responsibilities: [
 'Generate PDF exports',
 'Create Markdown files',
 'Generate shareable links',
 'Handle download processes',
]
}

```

```
 'Manage export states'
]
}
...
```

### ### Backend Components

#### #### GroqAPIClient

```typescript

```
interface GroqAPIClient {
  configuration: {
    apiKey: string;
    model: 'llama3-70b-8192';
    baseUrl: string;
  }

  methods: {
    createChatCompletion: (messages, options) => Promise<Response>;
    handleErrors: (error) => void;
    validateResponse: (response) => boolean;
  }

  responsibilities: [
    'Manage Groq API authentication',
    'Handle API requests/responses',
    'Implement error handling',
    'Manage rate limiting',
    'Validate API responses'
  ]
}
...
```

ContentProcessor

```typescript

```
interface ContentProcessor {
 methods: {
 parseTopicAnalysis: (response: string) => AnalysisData;
 extractFileContent: (file: File) => Promise<string>;
 fetchURLContent: (url: string) => Promise<string>;
 cleanHTMLContent: (html: string) => string;
 buildAnalysisPrompt: (content: string, type: string) => string;
 }

 responsibilities: [
 'Process different input types',
 'Extract and clean content',
 'Build AI prompts',
 'Parse AI responses',
 'Structure analysis data'
]
}
...
```

---

### ## Performance Considerations

### ### Frontend Optimization

- **Code Splitting**: Automatic with Next.js App Router
- **Image Optimization**: Built-in Next.js image optimization
- **CSS Optimization**: Tailwind CSS purging and minification
- **Bundle Analysis**: Webpack bundle analyzer integration
- **Caching**: Browser caching for static assets

### Backend Optimization

- **Serverless Functions**: Auto-scaling with Vercel/Firebase
- **API Response Caching**: Client-side response caching
- **Content Streaming**: Streaming responses for large analyses
- **Error Handling**: Graceful degradation and retry logic
- **Rate Limiting**: Client-side request throttling

### External Dependencies

- **Groq API**: High-performance inference with llama3-70b-8192
- **CDN Distribution**: Global content delivery
- **Edge Computing**: Edge function deployment for low latency

---

## ## Monitoring & Observability

### ### Error Tracking

```

```typescript
interface ErrorHandling {
  clientSide: [
    'React Error Boundaries',
    'Console error logging',
    'User-friendly error messages',
    'Fallback UI components'
  ]

  serverSide: [
    'API error responses',
    'Server-side logging',
    'Groq API error handling',
    'Request validation errors'
  ]
}
```

```

### ### Performance Monitoring

```

```typescript
interface PerformanceMetrics {
  frontend: [
    'Component render times',
    'Bundle size tracking',
    'Core Web Vitals',
    'User interaction metrics'
  ]

  backend: [
    'API response times',
    'Groq API latency',
    'File processing times',
    'Error rates'
  ]
}
```

```

...

---

## ## Future Architecture Considerations

### ### Scalability Enhancements

1. **Database Integration**: PostgreSQL/MongoDB for persistent storage
2. **User Authentication**: Auth0 or NextAuth.js integration
3. **Caching Layer**: Redis for response caching
4. **Load Balancing**: Multi-region deployment
5. **API Gateway**: Rate limiting and analytics

### ### Feature Extensions

1. **Real-time Collaboration**: WebSocket integration
2. **Advanced Analytics**: Usage tracking and insights
3. **AI Model Switching**: Support for multiple AI providers
4. **Enterprise Features**: Team management and permissions
5. **Integration APIs**: Third-party platform connections

### ### Security Hardening

1. **API Authentication**: JWT token implementation
2. **Data Encryption**: End-to-end encryption
3. **Audit Logging**: Comprehensive activity tracking
4. **Compliance**: GDPR/CCPA compliance features
5. **Security Headers**: Enhanced HTTP security headers

---

This architecture documentation provides a comprehensive overview of the GenAI-Scout Lite system, following C4 model principles and including detailed technical specifications for development, deployment, and future enhancement planning.