RAGSuite | PRD v1.0

Document Version: 1.0
`Last Updated: September 23, 2025
Status: Draft for Review

---

0) Scope Clarification — All-in-One App

- The platform is **one application** exposing both **Search** and **Chat** via the **same RAG pipeline**.
- Integrators can enable **search**, **chat**, or **both** per embed/config (mode: "search" | "chat" | "auto").

---

1) Executive Summary

1.1 Product Overview

The AI Search & Chatbot Platform is an API-first, self-hosted solution that enables organizations to deploy intelligent search and conversational AI across their digital properties. The platform combines automated website crawling, retrieval-augmented generation (RAG), and local large language models to provide contextually accurate responses based on organizational content.

1.2 Business Objectives

- **Primary Goal:** Deliver a production-ready AI search and chatbot solution for on-premises deployment
- **Market Position:** Cost-effective alternative to cloud-based AI solutions with complete data control
- **Target Users:** Organizations requiring private AI deployments with custom content integration

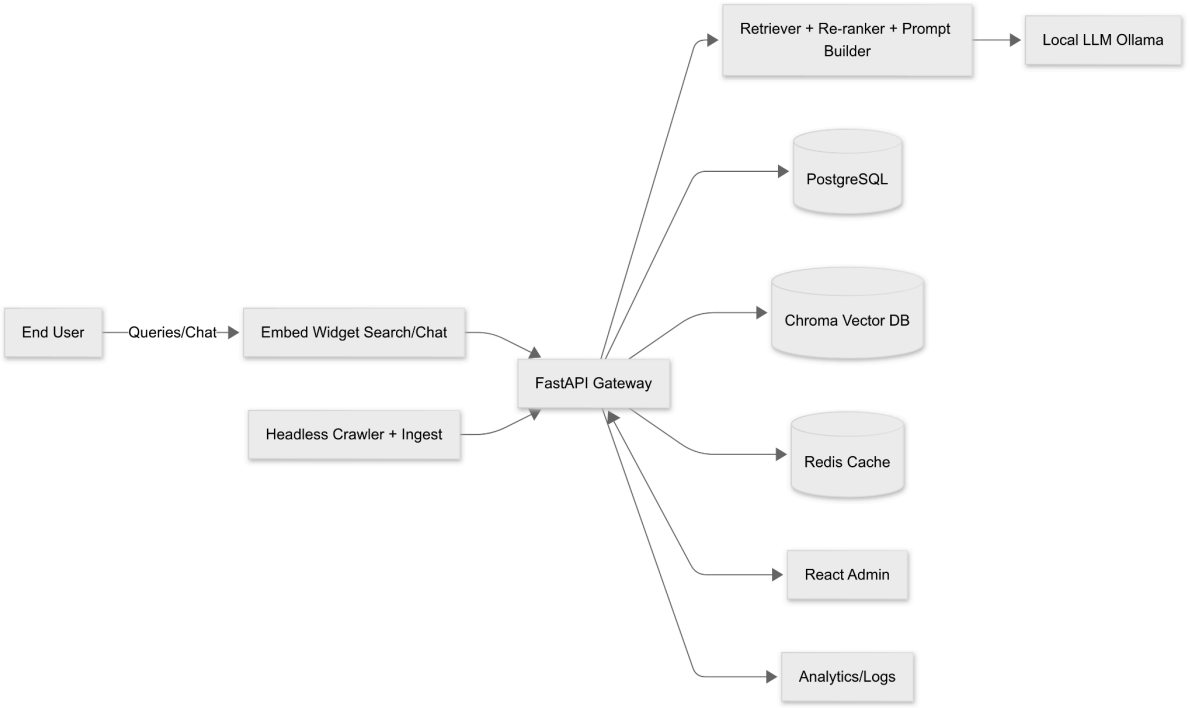
1.3 Success Metrics

- **Technical:** <10 second response times, 99% uptime during business hours
- **User Adoption:** 80%+ query satisfaction rate within first month of deployment
- **Business:** Complete v1 delivery within 16-week development cycle

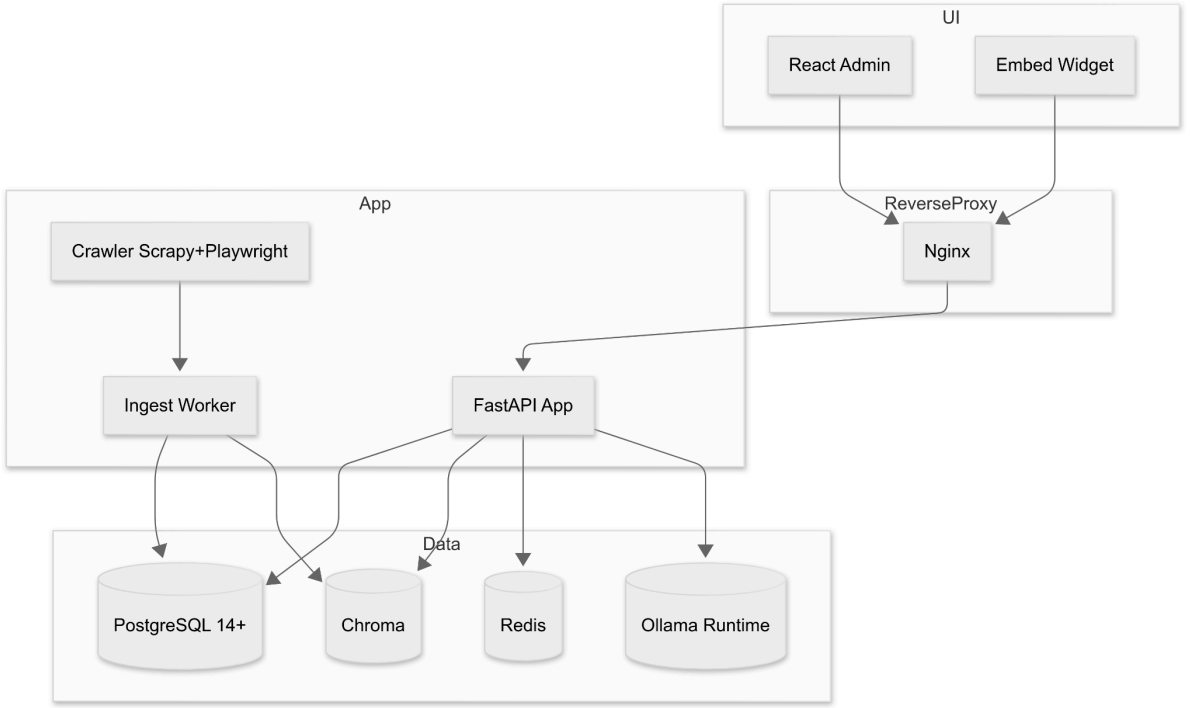
---

2) Product Architecture

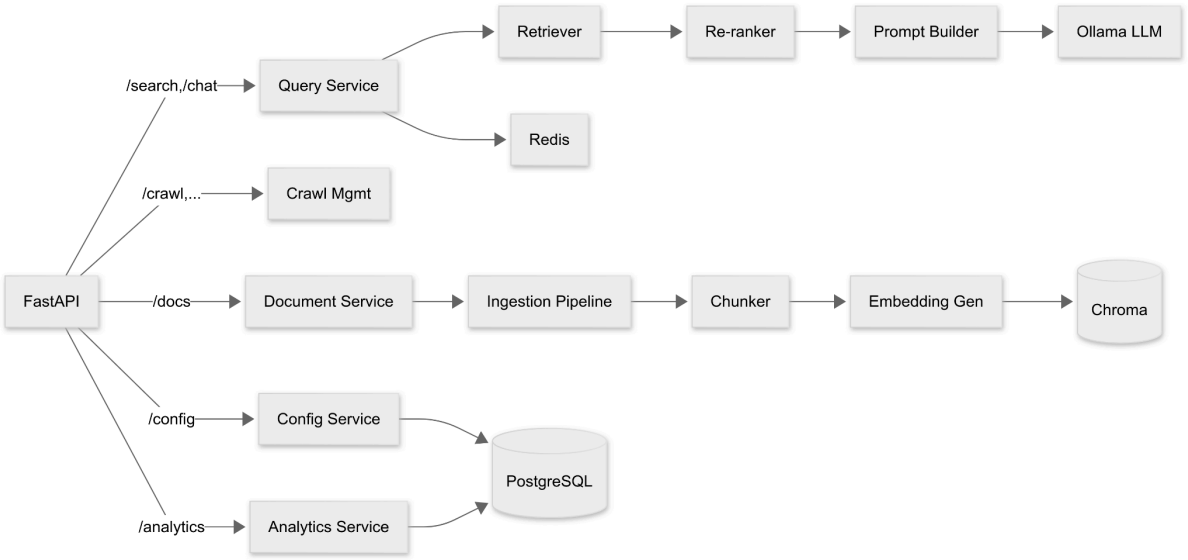
2.1 System Architecture (Context)



2.2 Container/Deployment Topology



2.3 Backend Components (C4-L3)



2.4 Technology Stack

Backend API Services

- **Framework:** FastAPI (Python 3.9+)
- **Database:** PostgreSQL 14+
- **Vector Database:** Chroma
- **Cache:** Redis
- **Web Crawler:** Scrapy with Playwright headless rendering
- **LLM Integration:** Ollama with Mistral 7B (default) or Llama 2 7B
- **Embeddings:** Sentence Transformers (all-MiniLM-L6-v2, 384-d)

Frontend Applications

- **Framework:** React 18+ with TypeScript
- **State Management:** Redux Toolkit or Zustand
- **UI Library:** Material-UI or Tailwind CSS
- **Build Tool:** Vite
- **API Client:** Axios with OpenAPI code generation

Infrastructure

- **Containerization:** Docker & Docker Compose
- **Process Management:** systemd or PM2 (optional)
- **Web Server:** Nginx (reverse proxy)
- **Monitoring:** Basic logging with structured JSON

---

3) API Specification

3.1 Core API Endpoints

Authentication & Configuration

- POST /api/v1/auth/login - Admin authentication
- GET /api/v1/config/system - System configuration
- PUT /api/v1/config/system - Update system settings
- GET /api/v1/config/themes - Available themes
- PUT /api/v1/config/branding - Update branding settings
- GET /api/v1/config/rag-defaults - Read RAG/reranker defaults

Content Management

- POST /api/v1/crawl/sites - Add crawling target
- GET /api/v1/crawl/sites - List crawling targets
- PUT /api/v1/crawl/sites/{id} - Update crawl configuration
- DELETE /api/v1/crawl/sites/{id} - Remove crawling target
- POST /api/v1/crawl/start/{id} - Start crawling job
- GET /api/v1/crawl/status/{id} - Check crawling status
- PUT /api/v1/crawl/preview - Return render+text sample for a URL (headless)

Document Management

- GET /api/v1/documents - List indexed documents
- POST /api/v1/documents/upload - Upload document manually
- PUT /api/v1/documents/{id} - Update document metadata
- DELETE /api/v1/documents/{id} - Remove document from index
- GET /api/v1/documents/{id}/content - Retrieve document content

Search & Chat Services

- POST /api/v1/search - Perform AI search query
- POST /api/v1/chat/message - Send chat message
- GET /api/v1/chat/sessions - List chat sessions
- DELETE /api/v1/chat/sessions/{id} - Clear chat session
- POST /api/v1/chat/feedback - Submit response feedback

Analytics & Monitoring

- GET /api/v1/analytics/overview - Dashboard metrics
- GET /api/v1/analytics/queries - Query statistics
- GET /api/v1/analytics/popular - Popular search terms
- GET /api/v1/health - System health check

3.2 API Response Format

```
{
  "success": true,
  "data": { },
  "message": "Operation completed successfully",
  "timestamp": "2025-09-22T10:30:00Z",
  "request_id": "uuid-string"
}
```

3.3 Error Handling

```
{
  "success": false,
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Invalid query parameters",
    "details": { }
  },
  "timestamp": "2025-09-22T10:30:00Z",
  "request_id": "uuid-string"
}
```

---

4) Feature Requirements

4.1 Administrator Dashboard (React Application)

Content Management Interface

- **Website Configuration**
  - URL input with validation
  - Crawl depth settings (1-5 levels)
  - Crawl frequency scheduling (hourly, daily, weekly)
  - Robots.txt compliance toggle
  - Sitemap.xml integration

- **Document Management**
    - File upload interface (PDF, DOC, DOCX, TXT, HTML)
    - Document preview and metadata editing
    - Bulk document operations
    - Index status monitoring

Search Configuration
- **Relevance Tuning**
    - Similarity score thresholds
    - Result ranking parameters
    - Response length controls
    - Citation formatting options
- **Content Filtering**
    - URL pattern inclusion/exclusion
    - Content type filtering
    - Language detection settings
    - Duplicate content handling

Analytics Dashboard
- **Usage Metrics**
    - Total queries processed
    - Average response times
    - Popular search terms
    - User session statistics
- **Performance Monitoring**
    - System resource usage
    - Crawling job status
    - Error rate tracking
    - Response quality metrics

Customization Panel
- **Theme Management**
    - Color scheme selection
    - Logo upload and positioning
    - Font family selection
    - Custom CSS injection
- **Branding Controls**
    - Application title and tagline
    - Footer customization
    - White-label options
    - Multi-language UI labels

4.2 Embedded Widget (React Component)

Search Interface
- **localQuery Input**
    - Natural language search box
    - Voice input support (browser API)

        - Search suggestions/autocomplete
        - Query history (session-based)
    - **Results Display**
        - Structured result cards
        - Source attribution with clickable links
        - Relevance score indicators
        - Result snippet highlighting

Chat Interface
- **Conversation Flow**
    - Message thread display
    - Typing indicators
    - Message timestamps
    - Context preservation across queries
- **Response Formatting**
    - Markdown support for rich text
    - Code syntax highlighting
    - Bullet point and numbered lists
    - Inline citations with hover previews

Widget Configuration
- **Embedding Options**
    - Multiple integration methods (iframe, script tag, React component)
    - Responsive design breakpoints
    - Z-index and positioning controls
    - Custom event callbacks
- **Styling System**
    - CSS custom properties
    - Theme inheritance from parent site
    - Mobile-first responsive design
    - Accessibility compliance (WCAG 2.1 AA)

---

5) Technical Requirements

API Response Times
- Search queries: <10 seconds (95th percentile)
- Chat responses: <10 seconds (95th percentile)
- Document upload: <30 seconds for files up to 50MB
- Dashboard loading: <3 seconds for initial load

Concurrent Usage
- Support for 50-100 concurrent users
- Queue management for crawling jobs
- Rate limiting: 100 requests per minute per API key
- Session timeout: 30 minutes of inactivity

Data Processing

- **Document Handling**
  - Maximum file size: 50MB per document
  - Supported formats: HTML, PDF, DOC, DOCX, TXT
  - Text extraction with metadata preservation
  - Automatic language detection
- **Content Indexing**
  - Document chunking: 512-1024 token segments
  - Embedding generation: 384-dimensional vectors
  - Incremental indexing with change detection
  - Duplicate content deduplication

Storage Requirements

- **Database Storage**
  - PostgreSQL for structured data
  - Estimated 1GB per 10,000 documents
  - Daily backup capabilities
  - Index optimization for search performance
- **Vector Storage**
  - Chroma vector database
  - Memory-mapped file storage
  - Automatic index rebuilding
  - Similarity search optimization

Security & Privacy

- **API Security**
  - API key-based authentication
  - Request rate limiting
  - Input validation and sanitization
  - SQL injection prevention
- **Data Privacy**
  - Local data processing only
  - No external API calls for LLM inference
  - Configurable data retention policies
  - User session data encryption

---

6) User Experience Requirements

6.1 Administrator User Journeys

Initial Setup Flow

1. **System Configuration**: Set up admin credentials; configure basic system settings; upload branding; test connectivity.

2. **Content Source Setup**: Add website URLs; configure crawling parameters; upload initial documents; verify indexing.
3. **Search Customization**: Adjust relevance parameters; test search; configure response formatting; set up analytics tracking.

Ongoing Management

1. **Content Maintenance**: Monitor crawl status; review/approve new content; update schedules; manage doc lifecycle.
2. **Performance Optimization**: Review analytics; tune relevance; optimize response times; monitor resources.

6.2 End-User Experience

Search Interaction

1. **Query Entry**: Natural language question input; real-time suggestions; voice input; easy edits.
2. **Result Consumption**: Scan relevant results; verify citations; follow-up queries; share results.

Chat Conversation

1. **Conversation Initiation**: Clear chat identification; welcome message; example queries; mode switching guidance.
2. **Ongoing Dialogue**: Natural follow-ups; context retention; source verification; history review.

---

7) Integration Requirements

7.1 Embedding Methods

- **JavaScript Widget**: CDN script, config via data attributes, events, minimal DOM footprint
- **React Component**: NPM package, TypeScript types, hooks API, SSR support
- **iframe Integration**: Cross-origin comms, responsive sizing, sandbox

7.2 API Integration

- **OpenAPI**: full docs, codegen, interactive testing, SDK generation
- **Authentication**: API key management, rotation, usage tracking, rate limit config

---

8) Deployment & Infrastructure

8.1 System Requirements

- **Minimum Hardware**: 4 cores, 8GB RAM (16GB recommended), 100GB SSD, outbound internet for setup
- **OS**: Ubuntu 20.04+ LTS
- **Prereqs**: Docker, Docker Compose, Python 3.9+, Node.js 18+

8.2 Installation Process

1. **Pre-deployment**: verify system; network; security baseline; backups
2. **Deployment**: compose up; DB init; model download; SSL; admin creation
3. **Post-deployment**: health checks; initial indexing; performance baseline

8.2.1 Installer & Docker Compose (files)

See https://drive.google.com/drive/u/0/folders/1tNawQBWMot5sS0t_yzdic0Lg-Ks_Kf7m

---

9) Success Criteria & Testing

9.1 Acceptance Criteria

- **Functional Requirements**
  - All API endpoints respond correctly
  - Search returns relevant results >80% of the time
  - Chat maintains context across conversation
  - Admin dashboard provides complete system control
- **Performance Requirements**
  - API responses under 10-second target
  - System handles 50 concurrent users
  - Crawling processes 10+ pages per minute
  - UI loads and responds within 3 seconds

9.2 Quality Assurance

- **Automated Testing**: API integration tests; React unit tests; E2E; perf benchmarking
- **Manual Testing**: cross-browser; mobile responsiveness; accessibility; security assessment

---

10) Development Timeline

10.1 Phase 1: Backend API Development

- Core FastAPI application setup
- Database schema and migrations
- Authentication and security layer
- Content crawling and processing APIs
- Search and chat service APIs

10.2 Phase 2: Frontend Application Development

- React admin dashboard development
- Embedded widget React components
- API integration and state management
- Responsive design implementation
- Theme and customization system

10.3 Phase 3: Integration & Testing

- End-to-end integration testing
- Performance optimization
- Security hardening
- Documentation completion
- Deployment automation and packaging

---

11) Future Roadmap (v2.0+)

- Multi-tenancy support for SaaS deployment
- Advanced analytics with custom reporting
- A/B testing framework for search optimization
- Advanced security features (RBAC, audit logs)
- Real-time collaboration features
- Webhooks; GraphQL API; OAuth/SAML; Third-party integrations

---

12) Personas & User Stories (with Acceptance Criteria)

12.1 Personas

- **Admin/Operator**: configures content sources, monitors health/analytics.
- **Integrator (Front-end Dev)**: embeds widget into sites/apps.
- **Knowledge Manager**: curates content, approves crawled docs, tunes relevance.

- **End User (Visitor/Agent)**: searches or chats to get answers fast.

12.2 Top User Stories (MoSCoW)

Admin/Operator

1. *MUST*: As an Admin, I can add a website to crawl with depth, cadence, and JS rendering toggle so that dynamic sites index correctly.
   AC: Given a valid URL, When I save, Then a test render preview and robots/sitemap validation appear; status transitions to READY.
2. *MUST*: As an Admin, I can see indexing health, error rates, and last crawl time per source.
   AC: Table with status badges; drill-down errors; export CSV.
3. *SHOULD*: As an Admin, I can tune similarity threshold and reranker top-k without redeploy.
   AC: Changes persist; new queries reflect settings; versioned config history.

Integrator

4. *MUST*: As an Integrator, I can embed the widget via <script> or React component with mode set to search/chat/auto.
   AC: Copy-paste examples; a sandbox page validates API key and connectivity.
5. *SHOULD*: As an Integrator, I can theme the widget to inherit brand tokens (colors, radius, fonts).
   AC: CSS variables applied; live preview.

Knowledge Manager

6. *MUST*: As a KM, I can upload PDFs/DOCs and edit metadata before indexing.
   AC: Upload > Preview > Edit > Index pipeline job visible with progress.
7. *SHOULD*: As a KM, I can de-duplicate near-identical content.
   AC: Suggested merges with similarity score; one-click dedupe.

End User

8. *MUST*: As a User, I can ask a question and get an answer with citations to original sources.
   AC: Response shows at least 3 citations when available; clicking opens source.
9. *MUST*: As a User, I can give thumbs-up/down with reason tags.
   AC: Feedback recorded; Admin analytics reflect aggregates.
10. *SHOULD*: As a User, I can switch between Search results and Chat answer in the same panel.
    AC: Tabbed UI or toggle; shared query state.

---

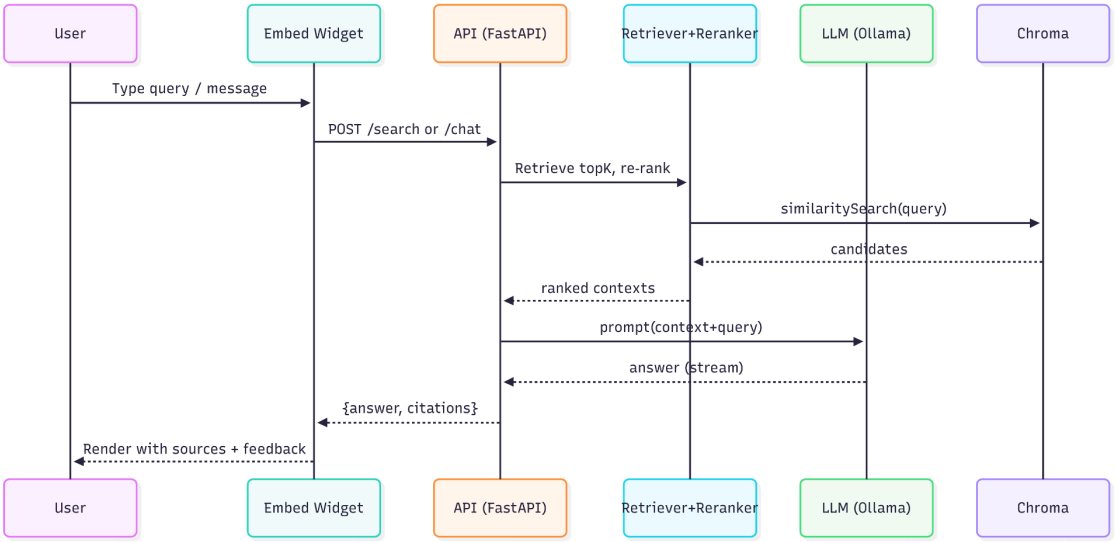
13) Risks & Mitigations

#	Risk	Likelihood	Impact	Mitigation	Owner	Trigger/Monitor
R1	Dynamic sites not captured by static crawler	Medium	High	Headless rendering via Playwright; preview endpoint; per-domain render budget	Backend TL	Crawl error rate >5% for JS pages
R2	Long LLM latencies on CPU-only hosts	Medium	High	Default to Mistral 7B, response streaming, caching; document HW sizing	Platform Lead	p95 latency >10s
R3	Vector DB bloat & slow recall	Medium	Medium	Chunk size per type, pruning policy, HNSW tuning, nightly compaction	Data Eng	Recall time >500ms or DB size growth >10%/wk
R4	Hallucinations on sparse corpora	Medium	High	Strict citation requirement, low temperature, reranker, claim-check heuristics	QA + NLP	Feedback downvotes >15%
R5	Compliance/privacy concerns	Low	High	Local inference only, configurable retention, PHI/PII filters, audit logs v2	Sec/Compliance	Periodic audits
R6	Rate-limit abuse on public widgets	Medium	Medium	Per-key quotas + IP throttling, CAPTCHA on burst, CDN caching	DevOps	429 spikes/DoS alerts
R7	Model/embedding version drift	Medium	Medium	Versioned index; dual-write and flip read alias; blue/green vectors	Data Eng	Model version change PR
R8	Single-node outages	Low	High	Compose profiles for HA, health checks, auto backups, documented restore	DevOps	Health probe failures
R9	Content license/copyright	Low	Medium	Respect robots, honor crawl allowlist, manual approval queue	KM Lead	Legal review flags

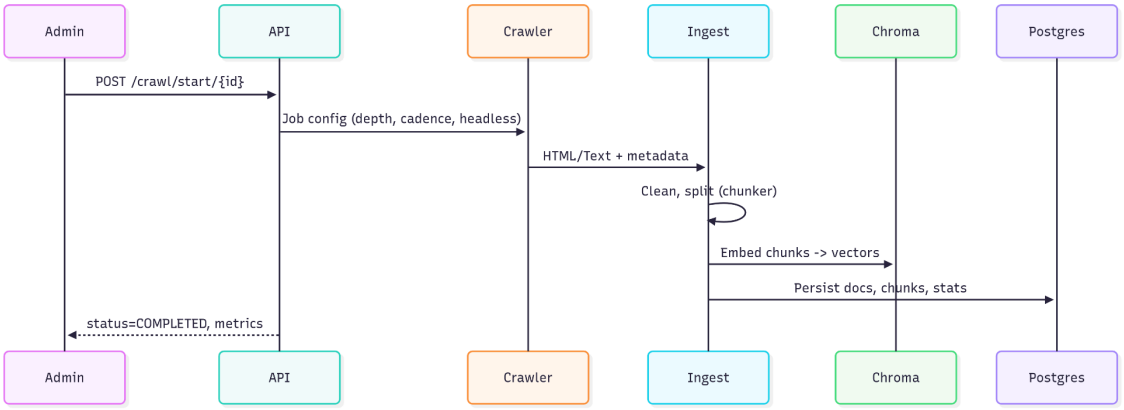
---

14) Sequence Diagrams

14.1 Search/Chat (unified)



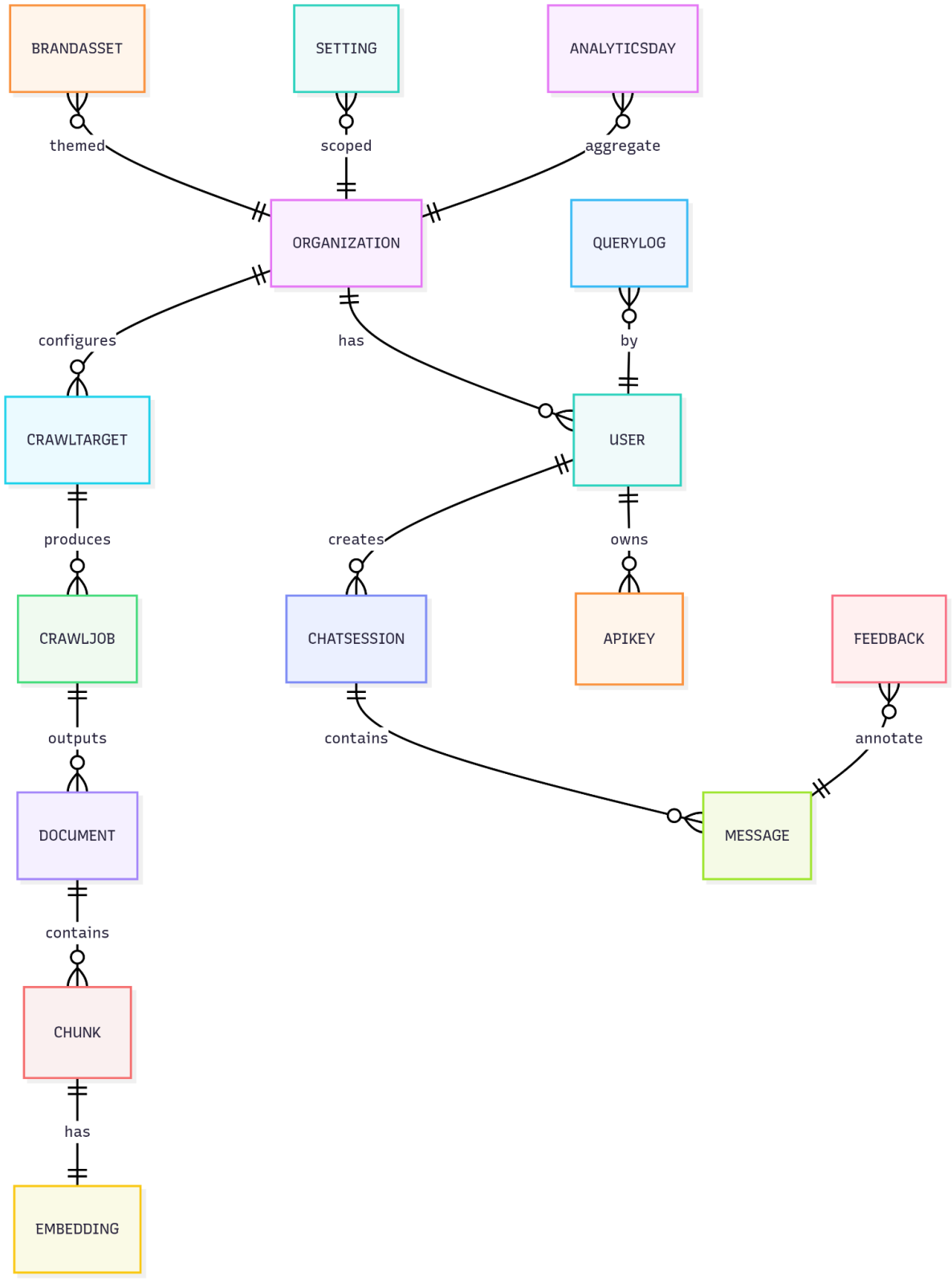
14.2 Crawl/Index



---

15) High-Level Data Model

15.1 ER Diagram



15.2 Key Fields (illustrative)

ORGANIZATION(id, name, plan, created_at)

USER(id, org_id, email, role[admin|km|viewer], password_hash, created_at)

APIKEY(id, org_id, key_hash, label, rate_limit_pm, created_at)

CRAWLTARGET(id, org_id, base_url, depth, cadence, headless[on|off|auto], allowlist, denylist, status)

CRAWLJOB(id, target_id, started_at, finished_at, pages_fetched, errors, status)

DOCUMENT(id, org_id, source_url, title, lang, checksum, status, last_indexed_at)

CHUNK(id, doc_id, ord, text, token_count, hash)

EMBEDDING(id, chunk_id, model, dim, vector_ref)

CHATSESSION(id, org_id, user_id|null, mode[search|chat|auto], started_at, meta)

MESSAGE(id, session_id, role[user|assistant|system], content, citations[], latency_ms, created_at)

QUERYLOG(id, org_id, apikey_id, query, mode, p95_latency, result_count, timestamp)

FEEDBACK(id, message_id, vote[up|down], reasons[], note, created_at)

ANALYTICSDAY(date, org_id, queries, avg_latency_ms, thumbs_up_rate, errors)

SETTING(id, org_id, key, value_json, version, updated_by)

BRANDASSET(id, org_id, logo_url, colors_json, css_overrides)

---

16) Defaults & Tunables (v1)

- **Embeddings**: sentence-transformers/all-MiniLM-L6-v2 (384-d)
- **Chunking**: HTML 800, PDF/DOC 600, MD/TXT 900; overlap 80
- **Retriever**: Chroma cosine; topK=24
- **Re-ranker**: BM25 default; optional mono-* (off by default)
- **Context Window**: top 10 chunks post-rerank (~2,500 tokens)
- **LLM**: mistral:7b default; llama2:7b alt; temperature 0.2; streaming on

- **Citations**: require ≥3 unique docs if available; fail closed if none
- **Dedup**: LSH on chunk hash; threshold 0.92
- **Cache**: Redis hash(query+config); TTL 10 min; stream-friendly
- **Headless Rendering**: Auto when low DOM text or missing selectors; 3s/page budget
- **Crawl Cadence**: Daily default; hourly/weekly per target; backoff on fails
- **Rate Limits**: 100 rpm/key; burst 20; 429 w/ Retry-After

---

17) Release, Canary & Rollback

- **Blue/Green** deploy; Nginx upstream points to active label
- **Canary**: route 10% via header X-Canary: 1 or IP hash
- **Promote** when error <1% and p95 <10s for 30 min
- **Rollback**: flip upstream; Alembic down for DB; vector dual-index alias
- **Feature Flags**: RERANKER_ON, STRICT_CITATIONS, HEADLESS_AUTO

---

18) Test Plan Additions

- **RAG Correctness**: 100 Q/A golden set; MRR/NDCG; cite coverage
- **Latency**: k6 50 VU / 10 min; assert p95 <10s
- **Crawler**: JS-heavy sample; preview vs indexed text similarity >0.8
- **Security**: OWASP Top-10; fuzz inputs
- **Resilience**: kill/restart containers; recover <30s

---

19) Backlog: GitHub/Jira (Owners & Timeboxes)

Roles: **BE** (Backend Lead), **FE**, **DE** (Data Eng), **DEV**, **QA**, **DO** (DevOps).
Two-week sprints ×8.

Epics

- **E1** Platform Foundations (Infra, API skeleton, Auth)n
- **E2** Crawl & Ingestion
- **E3** RAG/Search/Chat Core
- **E4** Admin App
- **E5** Embed Widget
- **E6** Analytics & Feedback
- **E7** Packaging, CI/CD, Installer
- **E8** QA, Perf, Security

Issues (sample)

Key	Title	Epic	Owner	Est.	Depends
P-1	FastAPI skeleton + auth + OpenAPI	E1	BE		—
P-2	DB schema + Alembic migrations	E1	BE		P-1
P-3	Redis cache scaffolding	E1	DEV		P-1
C-1	Crawler service (Scrapy+Playwright)	E2	DEV		P-2
C-2	Robots/sitemap resolver	E2	DEV		C-1
C-3	Crawl preview endpoint	E2	BE		C-1
I-1	Ingestion pipeline (clean/split)	E2	DE		P-2
I-2	Embedding generation + Chroma client	E2	DE		I-1
Q-1	Retriever + topK search	E3	DE		I-2
Q-2	Reranker (BM25/mono-*)	E3	DE		Q-1
Q-3	Prompt builder + citation rules	E3	BE		Q-2
Q-4	Chat endpoint (streaming)	E3	BE		Q-3
A-1	Admin shell (Vite+React+TS)	E4	FE		P-1
A-2	Content sources UI + preview	E4	DEV		C-3
A-3	Relevance tuning UI	E4	DEV		Q-2
A-4	Analytics dashboard	E4	DEV		E6
W-1	Widget (search/chat/auto)	E5	FE		Q-4
W-2	Theming (CSS vars) + docs	E5	DEV		W-1
X-1	Feedback API + UI	E6	BE		Q-4
X-2	Analytics ETL + PG views	E6	DE		X-1
PK-1	Docker Compose + Makefile	E7	BE		P-1
PK-2	Installer script + Nginx	E7	BE		PK-1

Key	Title	Epic	Owner	Est.	Depends
PK-3	Blue/green + canary doc	E7	BE		PK-2
QA-1	E2E tests (Playwright/Cypress)	E8	QA		E4,E5
QA-2	Load tests (k6)	E8	QA		Q-4
SEC-1	Security checklist	E8	QA		P-1

Definition of Done: code + tests + docs; CR passed; merged to main; CI green; feature behind flag if risky.

Sprint Cadence: 2-week sprints × 8; demo at Sprints 2/4/6; RC at Sprint 8.

---

20) Low-Fi Mockups / Prototype

20.1 Admin — Dashboard (ASCII)

20.2 Admin — Crawl Source Detail

URL: [https://example.com] Depth: [2] Cadence: [Daily]

Headless: [Auto] Robots: [✓] Sitemap: [✓]

[Preview Render] [Test Crawl] [Save]

Allowlist: *.example.com/docs/*

Denylist: *.example.com/private/*

Status: READY Last Crawl: 2025-09-22 22:10 Errors: 0

20.3 Widget — Unified Search/Chat

[Ask anything about ACME… (🎤)] (Search | Chat | Auto)

Answer: "Here's a summary…" [👍] [👎]

Sources: [Doc A] [Doc B] [Doc C]

[Show results tab] [Follow-up question…]

20.4 Embed Examples

```
<!-- Script tag (Auto mode) -->

<script src="/widget/embed.js"></script>

<script>

 AISearchChat.init({

  apiBase: "https://your-domain.example/api/v1",

  apiKey: "public-embeddable-key",

  mode: "auto",

  container: "#ai-widget",

  theme: { primary: "#0052cc", radius: 10 }

 });

</script>

<div id="ai-widget"></div>

// React (TypeScript)

<AISearchChatWidget
```

```
  apiBase="/api/v1"
  apiKey={env.PUBLIC_WIDGET_KEY}
  mode="search"
  onFeedback={(res, vote) => console.log({res, vote})}
/>
```

---

21) Admin & API Quickstart (Docs excerpt)

21.1 Admin

1. Install via install.sh.
2. Login /admin with default admin (change password).
3. Add **Crawl Target**; run preview; save & start crawl.
4. Upload a test PDF; verify documents appear.
5. Tune **RAG** thresholds; test queries in **Playground**.

21.2 API

POST /api/v1/search

```
{ "query": "Refund policy for EU?", "top_k": 10 }
```

Response includes: answer, citations[], latency_ms, request_id.

---

22) Open Questions

- What do we think about integrate MCP? *(Maybe in v2)*
- Default LLM for CPU-only: **mistral:7b**? *(Proposed: yes)*
- Tenancy: single-org in v1, multi-tenant in v2? *(Proposed: single-org v1)*

---

23) Document Approval

- ☐ Technical Architecture Review
- ☐ Product Management Approval
- ☐ Development Team Sign-off

- [ ] Quality Assurance Review
- [ ] Stakeholder Final Approval