Market & Technology Positioning

Elis AI Technical Documentation

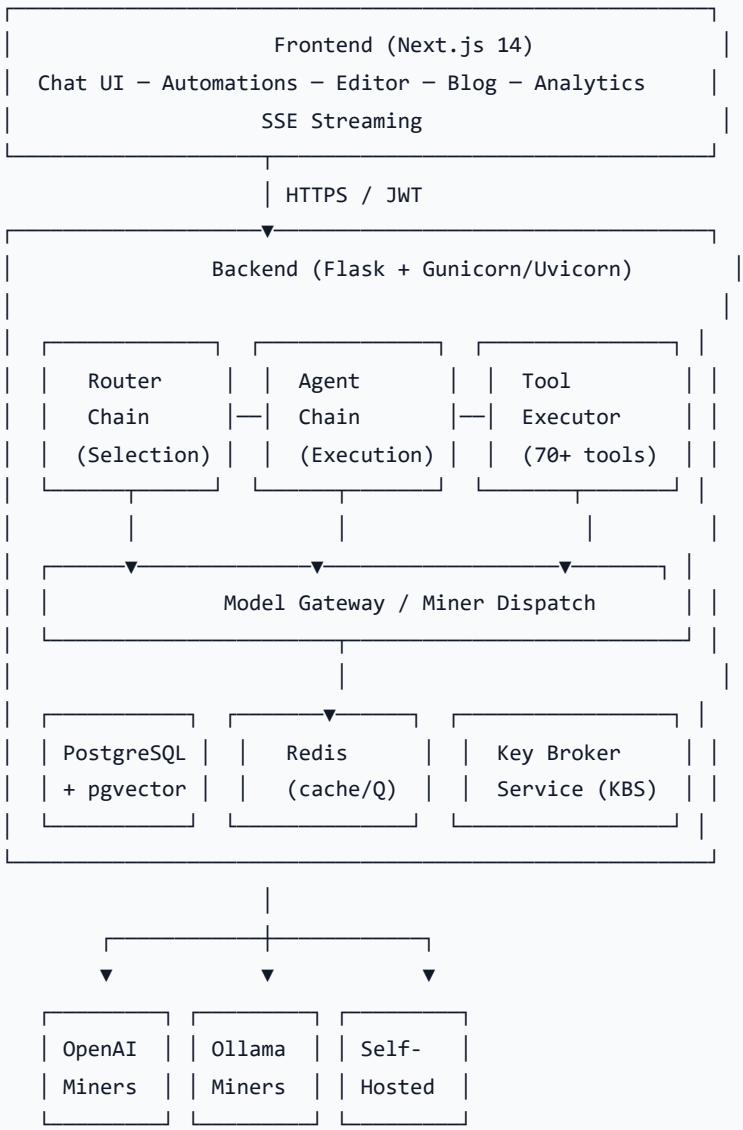
Generated: 2026-05-19 12:44 UTC

Repository: Guide Tech

Elis AI — Technical Guide

Architecture Overview

Elis AI is a dynamic AI orchestration platform built on two core chain layers — **router chains** (selection) and **agent chains** (execution) — with distributed model inference, a self-improving feedback loop, and full trace observability.



Tech Stack

Layer	Technology	Version
Frontend	Next.js, React, Tailwind CSS	14, 18, 3.x
Backend	Flask, Gunicorn, Uvicorn	Python 3.11+
Database	PostgreSQL + pgvector	16
Cache/Queue	Redis	7
AI Models	OpenAI (GPT-5 family), Ollama (Qwen, Phi, Gemma)	—
Embeddings	pgvector (multi-dimension: 256–4096)	—
NLP	spaCy, Presidio (PII), Microsoft NER	—
SQL Agent	LangChain	—
Browser	Selenium, Playwright	—
Encryption	Key Broker Service (envelope encryption)	—
Email	Azure Communication Services	—
Container	Docker Compose (local), Azure Container Apps (prod)	—

Core Systems

1. Router Chain (Selection Layer)

The router selects the optimal agent, model, and execution mode for each user prompt. **No hardcoded routes exist** — selection is entirely dynamic.

Decision pipeline:

```
User prompt
  │
  ▼
Embed prompt (vector representation)
  │
  ▼
Candidate retrieval (vector similarity search)
  ├── AgentRetriever     → ranked agent candidates
  ├── TemplateRetriever  → ranked template candidates
  ├── ToolBundleRetriever → ranked tool bundle candidates
  └── ModelProfileRetriever → ranked model candidates
  │
```

```
  ▼
Score & rank candidates
  ├── Similarity score (cosine distance)
  ├── Prior quality (EMA — exponential moving average)
  └── Live feasibility (miner available? tools functional? data source linked?)
  │
  ▼
Select: agent_id + model_id + miner_id + execution_mode
  │
  ▼
Emit router trace (decision metadata for audit + learning)
```

Execution modes:

Mode	Description	When Selected
`direct_answer`	Simple factual response, no tools	Low-complexity, factual queries
`retrieval_augmented`	Enhanced with KB/document retrieval	Queries matching uploaded documents
`skill_chain`	Multi-step: search → expertise → context → answer	Research-heavy queries
`multi_step_workflow`	Complex task decomposition with tool orchestration	Multi-part or ambiguous queries

Non-negotiable rule: Router selection uses similarity, quality history, and feasibility. No prompt-specific routing or static model pinning.

2. Agent Chain (Execution Layer)

Agents are reusable blueprints that define how a class of queries should be handled.

Agent definition:

```
Agent
├── system_prompt      — Domain-specific instructions
├── expert_domain      — Category (finance, research, general, etc.)
├── tool_bundle        — Curated set of tools this agent can use
├── execution_rules    — Constraints (max tool loops, token budget)
├── quality_metrics    — EMA scores from historical performance
└── examples           — Sample prompts for similarity matching
```

Execution flow:

```
Router output (agent + model + mode)
  │
  ▼
Build system prompt (agent instructions + context + data sources)
  │
  ▼
Execute tool loop
  ├── Model generates tool calls (function calling)
  ├── Tools execute and return results
  ├── Results are injected into conversation context
  ├── Loop continues until agent produces final answer or budget exhausted
  └── Complexity-scoped budgets control loop iterations
  │
  ▼
Post-process response
  ├── Extract citations and source URLs
  ├── Build evidence blocks
  ├── Format tables, charts, images
  └── Rehydrate redacted PII
  │
  ▼
Stream response via SSE
  │
  ▼
Emit trace packet (full execution audit)
```

3. Feedback & Learning Loop

Every execution is traced, graded, and used to improve future routing.

Trace packet contents:

- Router decision metadata (candidates, scores, selection reason)
- Agent selection and execution mode
- Tool calls with inputs/outputs and timing
- Model I/O with token counts
- Total latency breakdown
- Error events (if any)

Grading pipeline:

```
Response
  │
  ▼
ResponseEvaluator
  ├── Relevance (does it answer the question?)
  ├── Factual accuracy (validated via tools?)
  ├── Tool selection appropriateness
```

```
  ├── Clarity and completeness
  └── Source quality
  │
  ▼
Quality score (0.0 – 1.0)
  │
  ▼
EMA update propagation
  ├── Agent profile → influences future agent selection
  ├── Model profile → influences future model selection
  ├── Template candidates → promote successful patterns
  └── Tool bundle reliability → adjust tool availability
```

4. Miner System (Distributed Inference)

Models run on distributed worker containers called "miners."

Registration flow:

1. Miner starts and registers via WebSocket with enrollment secret.
2. Backend validates and adds to miner pool.
3. Regular health heartbeats confirm availability.
4. Tier classification: `nano`, `small`, `frontier`, `self-hosted`.

Dispatch flow:

```
Agent needs model inference
  │
  ▼
Model Gateway
  ├── Check model tier requirements
  ├── Filter available miners (healthy + matching model)
  ├── Score by reliability + latency
  └── Select best candidate
  │
  ▼
Dispatch to miner
  │
  ├── Success → return response
  └── Failure → reroute to next healthy miner (if capacity exists)
```

Model pair locking (consensus mode):

- For critical queries, two miners execute the same prompt.
- Responses are compared for consistency.
- If they agree → return result.
- If they disagree → flag for review or select higher-confidence response.

Tool Ecosystem (70+)

Web Research

Tool	Purpose
`web_search`	General web search
`web_search_discover`	Explore related topics
`web_content_reader`	Extract main content from a URL
`web_crawl`	Multi-page site crawl
`site_deep_dive`	Scrape structured data
`deep_search`	Multi-step retrieval with refinement
`trusted_web_search`	Fact-checked search with validation
`direct_url_lookup`	Bypass search, fetch URL directly

Browser Automation

Tool	Purpose
`browser_open_session`	Start Selenium/Playwright session
`browser_navigate`	Go to URL
`browser_type`	Fill form fields
`browser_click`	Click elements
`browser_read_dom`	Extract page HTML
`browser_screenshot`	Visual capture
`browser_network_traffic`	Intercept network requests
`browser_close_session`	Clean up

Data Collection

Tool	Purpose
`deep_collection`	Systematic property/listing collection
`save_collection_items`	Store structured data
`web_table_extractor`	Parse HTML/CSV tables
`extract_chart_data`	OCR charts and graphs
`merge_data_sources`	Combine multiple data streams

Database & SQL

Tool	Purpose
`SQL_agent_tool`	Natural-language SQL query execution (LangChain)
`sql_query`	Direct query with result limiting

Geographic & Spatial

Tool	Purpose
`geo_geocode`	Address → lat/lon
`geo_reverse`	lat/lon → address
`geo_normalize`	Standardize location formats
`geo_distance`	Calculate distances
`map_plot_points`	Render map with data points
`map_spatial_join`	Spatial analysis

Knowledge & Retrieval

Tool	Purpose
`conversation_retriever`	Recall earlier conversation turns
`document_query_retriever`	Search uploaded documents
`knowledge_base_service`	Search ingested knowledge base
`embedding_tool`	Vector similarity search
`evidence_retriever`	Find supporting sources

Content & Formatting

Tool	Purpose
`build_table_block`	Generate markdown tables
`build_chart_block`	Chart data structures
`generate_chart`	Create visualizations
`build_citation_block`	Format references
`html_to_markdown`	Convert HTML content

Validation & Trust

Tool	Purpose
`validate_claims`	Fact-check assertions
`contradiction_detector`	Find inconsistencies
`response_verifier`	Cross-check results
`link_safety_checker`	Detect malicious URLs

Specialized

Tool	Purpose
`rest_api_tool`	Call custom REST APIs
`decision_extractor`	Parse decisions from text
`constraint_extractor`	Identify requirements
`task_graph_builder`	Decompose complex tasks

Data Layer

PostgreSQL Schema (Key Tables)

```
agent_registry        — Agent blueprints (system prompt, tools, domain)
templates             — Agent templates with version history
model_registry        — Model catalog + performance metrics
conversations         — Multi-turn chat history
messages              — Individual messages + trace JSON
widget_runs           — Automation execution history
router_traces         — Routing decisions for learning
documents             — Uploaded documents metadata
companies             — Organization records
company_members       — Role assignments
data_sources          — SQL/API/doc connections
```

Embedding Tables (pgvector)

Table	Dimensions Supported
`agents_embeddings`	256, 384, 768, 1024, 1536, 2048, 3072, 4096
`templates_embeddings`	Same
`tool_bundles_embeddings`	Same
`documents_embeddings`	Same
`company_docs_embeddings`	Same

Dimension-specific columns avoid padding/truncation loss. The system selects dimension based on the embedding model tier in use.

Retrieval Pipeline

```
User query
  │
  ▼
Embed query (match dimension to active model)
  │
  ▼
Multi-stage search:
  1. Agent similarity (AgentRetriever)
  2. Template similarity (TemplateRetriever)
  3. Tool bundle similarity (ToolBundleRetriever)
  4. Model profile similarity (ModelProfileRetriever)
  5. Document similarity (HybridRetrieval: semantic + keyword)
  │
  ▼
Rank by relevance + feasibility
  │
  ▼
Return candidates to router/agent
```

Redis Usage

- Session caching (JWT sessions, temporary state)
- Message buffer (SSE streaming backpressure)
- Async task queue (automation scheduling, training jobs)

Security Architecture

Authentication

- JWT-based authentication for all API endpoints.
- Email verification required (configurable).
- Optional two-factor authentication.
- Token refresh and expiry management.

PII Redaction (Presidio)

```
User input → PII Detector (emails, phones, SSN, CC, addresses)
  │
  ├── Found → Replace with [REDACTED_TYPE] tokens
  │           Store mapping in redaction context
  │
  ▼
```

```
Send redacted text to model
  |
  ▼
Receive model response → Rehydrate [REDACTED_TYPE] → Final output
```

Data Encryption

- **At rest:** Envelope encryption via Key Broker Service (KBS).
- **In transit:** HTTPS for all client-backend communication.
- **Miner registration:** Enrollment secret required.
- **SQL credentials:** Encrypted before storage, decrypted only at query time.

Audit Logging

- Every API call logs: user, company, action, timestamp.
- AI traces persist full execution details.
- Admin can inspect any conversation or run.

Deployment

Local Development

```
docker-compose up -d --build
```

Services started:

Service	Port	Purpose
`frontend`	3000	Next.js UI
`backend`	8000	Flask API
`postgres`	5432	Primary database
`redis`	6379	Cache and queue
`kbs`	—	Key Broker Service
`pgadmin`	5050	Database admin UI
`miner-openai-*`	8080+	OpenAI model workers
`miner-ollama-*`	8081+	Ollama model workers

Production (Azure Container Apps)

```
Docker build → Push to Azure Container Registry
  │
  ▼
Azure Container Apps deploys:
  ├── Frontend container
  ├── Backend container
  ├── PostgreSQL (managed or container)
  ├── Redis (managed or container)
  ├── KBS container
  └── Miner containers (scaled per demand)
```

Key environment variables:

Variable	Purpose
`DATABASE_URL`	PostgreSQL connection string
`REDIS_URL`	Redis connection string
`JWT_SECRET`	Token signing key
`DB_ENCRYPTION_KEY`	Fernet key for envelope encryption
`MINER_ENROLLMENT_SECRET`	Miner registration auth
`OPENAI_API_KEY`	OpenAI model access
`REDACT_ENABLED`	Toggle PII redaction
`AZURE_COMMUNICATION_SERVICES_CONNECTION_STRING`	Email service
`AUTH_REQUIRE_EMAIL_VERIFICATION`	Email verification toggle

Scaling

- **Horizontal:** Add miner containers for more inference capacity.
- **Vertical:** Increase container resources for backend or database.
- **Queue-based:** Redis absorbs burst traffic; workers process asynchronously.
- **Health-based:** Degraded miners are automatically removed from the pool.

API Overview

Core Endpoints

Method	Path	Purpose
`POST`	`/api/auth/register`	Create account
`POST`	`/api/auth/login`	Authenticate
`POST`	`/api/conversations`	Create conversation
`POST`	`/api/conversations/{id}/messages`	Send message
`GET`	`/api/conversations/{id}/messages`	Get messages (SSE stream)
`GET`	`/api/conversations/{id}/trace`	Get execution trace
`POST`	`/api/automations`	Create automation widget
`POST`	`/api/automations/{id}/run`	Trigger manual run
`GET`	`/api/automations/{id}/runs`	Get run history
`POST`	`/api/datasources`	Add data source
`GET`	`/api/agents`	List available agents
`GET`	`/api/models`	List available models
`GET`	`/api/analytics/runs`	Query run analytics

SSE Streaming Protocol

Responses stream via Server-Sent Events:

```
event: token
data: {"content": "The", "index": 0}

event: token
data: {"content": " answer", "index": 1}

event: source
data: {"url": "https://...", "title": "..."}

event: tool_call
data: {"tool": "web_search", "input": "...", "output": "..."}

event: done
data: {"trace_id": "abc-123", "tokens_used": 847}
```

Debugging & Diagnostics

Trace Inspection

Every run produces a trace accessible via API or the admin UI:

```
{
  "trace_id": "abc-123",
  "router": {
    "candidates": [...],
    "selected_agent": "financial-analyst-v3",
    "selected_model": "gpt-5",
    "confidence": 0.92,
    "execution_mode": "skill_chain"
  },
  "tool_calls": [
    {"tool": "web_search", "input": "...", "output": "...", "latency_ms": 340},
    {"tool": "build_table_block", "input": "...", "output": "...", "latency_ms": 12}
  ],
  "model_io": {
    "input_tokens": 2400,
    "output_tokens": 847,
    "model": "gpt-5",
    "miner_id": "miner-openai-1"
  },
  "total_latency_ms": 3200,
  "quality_score": 0.88
}
```

Diagnostic Checklist

When diagnosing issues, validate the full path in order:

1. **Auth** — Is the user authenticated? Valid JWT?
2. **Conversation** — Does the conversation exist? Correct company scope?
3. **Message** — Was the message persisted? Is it in the correct format?
4. **Router** — Did the router select an agent? Check trace for candidates and scores.
5. **Agent** — Did the agent execute? Check tool calls and model I/O.
6. **Trace** — Is the trace complete? All fields populated?
7. **Response** — Is the response non-empty? Does it answer the question?

Health Checks

- Backend: `GET /api/health`
- Miner: WebSocket heartbeat (periodic ping)

- Database: Connection pool monitoring
- Redis: `PING` command

Log Inspection

```
# Backend logs
docker logs backend --tail 200

# Miner logs
docker logs miner-openai-1 --tail 200

# Database logs
docker logs postgres --tail 200
```

Development Workflow

Adding a New Tool

1. Define the tool function in the tool executor module.
2. Register it in the tool registry with name, description, and parameters.
3. Add to relevant tool bundles (or create a new bundle).
4. Embed the tool bundle description for similarity matching.
5. Test with a probe prompt that should trigger the tool.
6. Verify the tool appears in trace output.

Adding a New Agent

1. Define agent blueprint: system prompt, expert domain, tool bundle, examples.
2. Insert into `agent_registry` table.
3. Embed the agent description and examples.
4. Router will automatically consider it for matching prompts.
5. Monitor quality scores after deployment.

Testing

- **Single-prompt probes:** Test one question, inspect trace.
- **10-industry benchmark:** Automated evaluation across verticals.

- **Automation tests:** Validate widget scheduling and report generation.
- **E2E tests:** Browser-based tests for full user flows.

Deployment Checklist

1. Run tests locally.
2. Build Docker images.
3. Push to container registry.
4. Deploy to Azure Container Apps.
5. Verify health endpoints.
6. Run a single-prompt probe to confirm end-to-end.
7. Monitor dashboards for anomalies.

Repository: Guide Sales & Marketing

Elis AI — Sales & Marketing Guide

What Is Elis AI?

Elis AI is an enterprise AI research platform that automates knowledge-intensive work. It intelligently routes queries across multiple AI models, executes multi-step research workflows with 70+ specialized tools, and learns from every interaction to deliver progressively better results.

Think of it as a tireless research team that searches the web, queries databases, collects structured data, generates reports, and publishes content — on a schedule, at scale, with full transparency into how every answer was produced.

Positioning Statement

For knowledge workers and research teams who spend hours gathering, synthesizing, and reporting on data across scattered sources, **Elis AI is an intelligent research automation platform** that dynamically orchestrates AI models, tools, and data sources to deliver accurate, evidence-backed answers in real time — unlike generic chatbots, Elis adapts to each query, learns from feedback, and automates recurring research tasks.

Key Value Propositions

1. Research Automation at Scale

- Users define recurring research tasks ("widgets") that run on a schedule — daily, weekly, or on-demand.
- Each widget produces structured reports with charts, tables, citations, and delta-first updates (only new data since the last run).
- **Pitch angle:** "Turn a 4-hour weekly research task into a 30-second automation."

2. Multi-Source Intelligence in One Platform

- Elis searches the web, queries connected SQL databases, reads uploaded documents, calls REST APIs, and browses live websites — all within a single conversation.
- 70+ specialized tools are selected automatically based on the query.
- **Pitch angle:** "One question, every source — no tab-switching, no copy-paste."

3. Self-Improving AI That Gets Smarter Over Time

- Every conversation is traced, graded, and fed back into the routing system.
- Agents that perform well are promoted; underperforming pathways are demoted.
- **Pitch angle:** "The more your team uses Elis, the better it gets — no manual tuning required."

4. Full Transparency & Trust

- Every response includes source citations with clickable links.
- A trace inspector shows exactly which tools were called, which model was used, and how the answer was assembled.
- **Pitch angle:** "No black boxes. See the reasoning behind every answer."

5. Enterprise-Grade Privacy & Compliance

- Built-in PII detection and redaction (Presidio) before any data reaches an AI model.
- Data is rehydrated after inference — sensitive fields never leave the perimeter unprotected.
- Multi-tenant isolation, role-based access, and full audit logging.
- **Pitch angle:** "Your data stays yours. Redaction, encryption, and audit trails are built in."

Target Industries & Use Cases

Industry	Example Use Case	Tools Involved
Finance & Investment	Automated daily earnings digest across 20 companies	Web search, table extraction, chart generation
Healthcare & Pharma	Drug pipeline competitor tracking	Deep search, data collection, scheduled reporting
Real Estate	Weekly market vacancy and REIT performance reports	Web crawl, SQL queries, geocoding, map plots
Retail & Ecommerce	Competitor pricing and product analysis	Browser automation, structured data extraction
Logistics & Supply Chain	Carrier and 3PL benchmarking	REST API integration, data merge, table building
Enterprise Research	M&A due diligence and vendor evaluation	Multi-step workflows, document retrieval, citation

---

Competitive Differentiation

Capability	Generic Chatbots	Elis AI
Multi-model orchestration	Single model	Dynamic routing across OpenAI, Ollama, self-hosted
Tool selection	Limited or manual	70+ tools, auto-selected per query
Recurring automation	Not available	Scheduled widgets with structured reports
Feedback-driven learning	Static	Quality scores improve routing over time
Trace & observability	Black box	Full audit trail for every response
Data source integration	Chat-only	SQL, documents, APIs, web crawl — all native
PII redaction	Optional/manual	Automatic detection and masking before inference

Demo Scenarios That Win Deals

Scenario 1: "The Instant Analyst"

1. Ask Elis a complex financial question (e.g., "Compare P/E ratios of the top 5 semiconductor companies and their 12-month outlook").
2. Watch it execute web search → table extraction → chart generation → synthesized answer with citations.
3. Show the trace inspector to prove every data point is sourced.

Scenario 2: "Set It and Forget It"

1. Create an automation widget: "Weekly commercial real estate vacancy report for Houston."
2. Schedule it for every Monday at 8 AM.
3. Show the report output: charts, tables, delta-first updates, clickable sources.
4. Emphasize: "This runs every week, automatically, with no human intervention."

Scenario 3: "Bring Your Own Data"

1. Connect a SQL database (e.g., CRM or inventory system).
2. Ask a natural-language question against the data ("Which accounts had the highest churn last quarter?").

3. Show the SQL query Elis generated, the results table, and the narrative summary.
4. Highlight: credentials are redacted, queries are sandboxed.

Scenario 4: "From Chat to Blog in 60 Seconds"

1. Have a research conversation about an industry trend.
2. Open the Editor, let AI rewrite the highlights.
3. Publish directly to the company blog with audience targeting.
4. Show the training pipeline generating related content topics automatically.

Messaging Framework

Headline Options

- "AI-Powered Research Automation for Knowledge Teams"
- "Turn Hours of Research into Minutes — Automatically"
- "The Research Platform That Gets Smarter Every Day"

Elevator Pitch (30 seconds)

> Elis AI is a research automation platform that connects to your data sources, searches the web, and generates evidence-backed reports — all orchestrated by AI that learns from every interaction. Teams use it to automate recurring research, cut analyst hours, and make better decisions faster.

For the C-Suite (10 seconds)

> Elis turns manual research workflows into automated, AI-driven intelligence pipelines — with full auditability and enterprise security.

Content Marketing Angles

Blog Topics

- "How AI Orchestration Outperforms Single-Model Chatbots"
- "5 Research Tasks Every Analyst Should Automate This Quarter"

- "The Hidden Cost of Manual Research in [Industry]"
- "Why Trace Transparency Matters for Enterprise AI Adoption"
- "From Chat to Report: How Recurring Research Automation Works"

Lead Magnets

- **ROI Calculator:** Input hours spent on recurring research → output time and cost savings.
- **Industry Benchmark Report:** Show Elis performance across 10 verticals (built-in benchmark system).
- **Whitepaper:** "Dynamic AI Orchestration: Why One Model Isn't Enough."

Social Proof Metrics

- 70+ specialized AI tools
- 10-industry benchmark evaluation
- Multi-model orchestration (not locked to one provider)
- Built-in PII redaction and audit trails
- Sub-second response streaming

Pricing & Packaging Signals

(Adjust based on actual pricing strategy)

Tier	Audience	Key Features
Starter	Individual researchers	Chat, web search, document upload, 5 automations
Team	Small teams (5–20)	All Starter + SQL integration, shared automations, role management
Enterprise	Large orgs (20+)	All Team + SSO, custom agents, API access, dedicated miners, audit logs

---

Objection Handling

Objection	Response
"We already use ChatGPT/Copilot."	Elis orchestrates across multiple models and tools — it selects the best AI for each task, connects to your databases, and automates recurring work. Generic chatbots don't do that.
"How do we trust the answers?"	Every response has a trace inspector showing sources, tool calls, and reasoning. Plus, built-in fact-checking tools validate claims before they reach you.
"What about data privacy?"	PII is automatically detected and redacted before reaching any model. Credentials are encrypted. Full audit logging tracks every action.
"We don't have technical staff to set this up."	The platform runs as a managed service. Data sources connect via UI. Automations are created through conversation — no code required.
"How is this different from building our own RAG pipeline?"	Elis includes the router, agents, tools, feedback loop, vector store, UI, and automation layer out of the box. Building this internally is 6–12 months of engineering.

Repository: Guide Business Operations

Elis AI — Business & Operations Guide

Platform Overview

Elis AI is an AI-powered research and automation platform built for organizations that rely on data-driven decision-making. It consolidates web research, database queries, document analysis, and recurring reporting into a single system — governed by role-based access, audit trails, and multi-tenant isolation.

This guide covers the operational capabilities, organizational governance, and day-to-day workflows relevant to business leaders and operations teams.

Organizational Structure

Multi-Tenant Architecture

Every organization operates in an isolated environment. Data, conversations, agents, automations, and data sources are scoped to each company. No cross-tenant data leakage is possible.

```
Organization (Company)
├── Members (Owner, Admin, Member, Viewer)
├── Conversations (chat history per user)
├── Data Sources (SQL, documents, APIs)
├── Automations (scheduled research widgets)
├── Agents (custom or system-provided AI blueprints)
├── Blog (auto-published content)
└── Audit Log (full action history)
```

Role-Based Access Control

Role	Capabilities
Owner	Full control — billing, member management, delete organization
Admin	Manage members, data sources, automations, agents, settings
Member	Create conversations, run automations, upload documents, use editor
Viewer	Read-only access to conversations and reports

Member Management

- Invite via email with role assignment
- Email verification required (configurable)
- Optional two-factor authentication
- Member activity visible in audit log
- Transfer ownership between users

Core Operational Workflows

1. Research Conversations

Users interact with Elis through a chat interface. Each conversation maintains context across multiple turns, allowing follow-up questions, refinements, and deep dives.

What happens behind the scenes:
1. User sends a message.
2. The **router** evaluates the query against available agents, models, and data sources.
3. The best **agent** is selected (e.g., a financial analyst agent, a web research agent).
4. The agent executes a **tool chain** — searching the web, querying databases, reading documents.
5. Results are synthesized into a response with **citations and evidence**.
6. The full execution is **traced** for audit and quality improvement.

Operational value: Every answer is reproducible and auditable. The trace shows which sources were consulted, which tools were used, and how the answer was assembled.

2. Automations (Recurring Research)

Automations are the operational backbone of the platform. They turn one-time research into scheduled, repeatable workflows.

Key capabilities:
- **Create from conversation:** Turn any successful research chat into a recurring automation.
- **Create from scratch:** Define a research task, select a schedule, configure data sources.
- **Schedule options:** Daily, weekly, or manual trigger.
- **Output format:** Multi-page reports with tables, charts, citations, and delta-first updates.
- **Run history:** Every execution is logged with results, timing, and status.

Automation lifecycle:

Stage	Description
Idle	Waiting for next scheduled run
Running	Currently executing the research workflow
Completed	Report generated and available for review
Error	Execution failed — check logs for details

Grid dashboard:
- Visual layout of all automations as cards
- Status badges (idle, running, error)
- Last run time and next scheduled run
- Quick actions: run now, edit, clone, delete
- Drag-and-drop reordering
- Responsive layout across device sizes

3. Data Source Management

Elis connects to external data sources to enrich research with proprietary information.

Supported sources:

Source Type	Description	Setup
SQL Databases	MySQL, PostgreSQL, MSSQL — natural-language queries	Connection string + credentials (encrypted)
Documents	PDFs, Word, text files — chunked, embedded, and searchable	Upload via UI
REST APIs	Custom endpoints — call with redacted auth headers	URL + headers + method
Google Docs	OAuth integration — import and index	Google OAuth flow

Security controls:
- Credentials are encrypted at rest (envelope encryption via Key Broker Service).
- SQL queries are sandboxed with result limiting.
- Auth headers and API keys are redacted in traces.
- Connectivity is tested before activation.

Linking to conversations:
Data sources can be linked to specific conversations or projects. When linked, the AI restricts its

search scope to the linked sources — ensuring relevance and preventing data spillover.

4. Document & Knowledge Base

Organizations build a searchable knowledge base by uploading documents.

Ingestion pipeline:
1. Upload document (PDF, DOCX, TXT, etc.)
2. Extraction — text is pulled from the document.
3. Chunking — text is split into semantic segments.
4. Embedding — each chunk is vectorized for similarity search.
5. Indexing — stored in pgvector for fast retrieval.

Usage:
- Documents are automatically consulted when a user asks a question relevant to the uploaded content.
- The retrieval system uses hybrid search (semantic similarity + keyword matching).
- Source citations link back to the original document and page.

5. Editor & Publishing

The built-in editor allows users to draft, refine, and publish content.

Capabilities:
- Markdown and plaintext editing
- AI-powered rewrite suggestions
- Insert data from linked sources
- Publish to the organization's blog
- Audience targeting (select who sees the content)
- Version history with diff comparison

Operational use cases:
- Weekly market intelligence briefings
- Internal research summaries
- Client-facing reports
- Auto-generated content from the training pipeline

6. Blog (Automated Content)

The blog module publishes research-derived content for internal or external audiences.

- Auto-generated from the training pipeline (scheduled topic generation)
- Audience segmentation (Sales, Business, Operations, Tech, Marketing)

- Category and tag organization
- Public access (no auth required for readers)
- Admin CRUD for manual curation

Governance & Compliance

Audit Trail

Every action in the platform is logged:
- Who performed the action (user, role, company)
- What was done (conversation, automation, data source change)
- When it happened (timestamp)
- Full trace for AI-generated content (router decision, tools used, model selected)

PII Protection

- **Automatic detection:** Emails, phone numbers, SSNs, credit cards, addresses, and other PII are identified using Microsoft Presidio.
- **Redaction:** Sensitive data is masked before being sent to any AI model.
- **Rehydration:** After the model responds, original data is restored in the output.
- **Configurable:** Redaction can be enabled/disabled via environment settings.

Data Isolation

- All data is scoped to the organization.
- No cross-tenant queries or access.
- Database-level isolation for SQL sources.
- Document embeddings are company-scoped.

Encryption

- Credentials encrypted at rest via Key Broker Service (envelope encryption).
- JWT-based authentication for all API calls.
- HTTPS in transit for all communication.

Monitoring & Analytics

Built-In Dashboards

Dashboard	What It Shows
Runs Browser	All AI executions, filterable by agent, model, status, time
Agent Quality	EMA scores, success rates, usage counts per agent
Automation History	Run logs for each widget — timing, status, outputs
Token Usage	Tokens consumed per user, per model, per time period
Model Reliability	Uptime and failure rates per model tier
Miner Health	Status of distributed inference workers
Anomaly Detection	Flagged unusual patterns (sudden failures, quality drops)

Key Operational Metrics

Metric	Description	Why It Matters
Research task latency	Time from query to complete response	Measures user experience
Automation success rate	% of scheduled runs that complete without error	Measures reliability
Agent selection accuracy	How often the router picks the best agent	Measures orchestration quality
Tool call success rate	% of tool invocations that return valid results	Measures tool ecosystem health
Feedback loop velocity	How quickly quality scores update after grading	Measures learning speed

---

Operational Scenarios

Scenario: Onboarding a New Team

1. **Create organization** — Owner signs up, names the company.
2. **Invite members** — Admin sends email invites with role assignments.
3. **Connect data sources** — Link SQL databases, upload key documents.
4. **Set up automations** — Define 3–5 recurring research tasks.

5. **Configure agents** — Optionally create custom agents with domain-specific prompts.
6. **Monitor** — Use dashboards to track adoption, quality, and usage.

Scenario: Weekly Market Intelligence Workflow

Step	Who	Action
1	Admin	Creates automation: "Weekly competitor pricing analysis"
2	System	Runs every Monday at 7 AM — searches web, queries SQL, builds tables
3	System	Generates report with charts, tables, delta-first updates
4	Analyst	Reviews report, asks follow-up questions in conversation
5	Analyst	Uses editor to draft client-facing summary
6	Admin	Publishes to internal blog for team distribution

Scenario: Connecting a SQL Database

1. Navigate to **Data Sources** panel.
2. Select **Add SQL Database**.
3. Enter connection details (host, port, database, username, password).
4. System tests connectivity and validates access.
5. Credentials are encrypted and stored securely.
6. Link the source to specific conversations or make it org-wide.
7. Users can now ask natural-language questions against the database.

Scenario: Investigating an Anomaly

1. Anomaly detection flags a sudden quality drop for a specific agent.
2. Admin opens **Runs Browser**, filters by that agent.
3. Inspects traces — sees a tool (web_search) returning empty results.
4. Identifies root cause: a website changed its structure.
5. Files a report; system re-routes to alternative tools until fixed.

Cost & Resource Management

Model Tiers

Tier	Examples	Cost Profile	When Used
Frontier	GPT-5	Highest quality, highest cost	Complex multi-step research
Small	GPT-4.1-mini	Good quality, moderate cost	Standard queries
Nano	Qwen, Phi, Gemma	Lower cost, self-hosted	Simple lookups, bulk tasks
Self-Hosted	Ollama models	Infrastructure cost only	Privacy-sensitive, high-volume

Cost Controls

- Model tier selection is automatic but influenced by query complexity.
- Token budgets can be set per request.
- Automations can be configured to use cost-efficient tiers.
- Analytics dashboard shows token consumption per user and model.

Capacity Planning

- Miner workers scale horizontally (add more containers).
- Redis handles message queuing for burst traffic.
- PostgreSQL with pgvector handles embedding storage.
- Health checks automatically remove degraded miners from the pool.

Integration Points

Integration	Direction	Purpose
SQL Databases	Inbound	Query proprietary data via natural language
REST APIs	Inbound/Outbound	Connect external services
Google Docs	Inbound	Import and index documents
Document Upload	Inbound	PDFs, Word, text for knowledge base
Blog	Outbound	Publish research to internal/external audiences
Email (Azure)	Outbound	Invitations, notifications, verification
SSE Streaming	Outbound	Real-time response delivery to clients

Business Continuity

- **Data persistence:** PostgreSQL with standard backup protocols.
- **Cache resilience:** Redis with configurable persistence.
- **Miner failover:** Automatic reroute to healthy miners on failure.
- **Audit recovery:** Full trace logs enable reconstruction of any historical response.
- **Multi-model redundancy:** If one model provider is down, the router selects an alternative.

Repository: Company Expansion

Plan: Miner Packaging, SQL Agent, Role Expansion, Isolation Validation

TL;DR

Six workstreams: (A) package the miner as an obfuscated downloadable with a dev setup guide, (B) add per-company SQL database connections using LangChain SQL agent, (C) expand company roles to user/admin/dev with feature-gating, (D) validate PII redaction, (E) validate company/project artifact isolation, (F) validate company agent access control. The codebase already has Cython obfuscation via `miner/Dockerfile.obfuscated`, a full privacy/redaction pipeline, and company-scoped RAG — but LangChain doesn't exist yet, roles need a "dev" tier, and the chat route has a membership re-check gap.

What Exists Today (relevant to this plan)

Miner System ✅

- `miner/app.py` — Flask server with `/generate`, `/generate-secure`, auto-enroll via WS
- `miner/Dockerfile.obfuscated` — Cython multi-stage build that strips .py source, keeps .so
- `miner/.env` / `.env.azure` — env vars: BACKEND_URL, MINER_ENROLLMENT_SECRET, OPENAI_API_KEY, MODEL_NAME
- `backend/miner_api.py` — 17 endpoints (register, auto-enroll, claim, disable, quarantine, etc.)
- `backend/miner_ws.py` — WebSocket connection manager with heartbeat
- `backend/miner_hardening.py` — HMAC verification, rate limiters, timestamp freshness

Privacy/Redaction ✅

- `backend/agents/privacy_policy.py` — 3-zone policy (SAFE, REDACT_FOR_MINER, REDACT_ALWAYS)
- `backend/agents/redactor.py` — NER-based detection, placeholder generation, rehydration
- `backend/agents/privacy_context.py` — Request-scoped PrivacyContext threaded through execution
- `backend/agents/tool_privacy.py` — Field-level tool argument redaction
- Controlled by `REDACT_ENABLED` and `REDACT_NER_ENABLED` env vars

Company Isolation ⚠️

- `backend/agents/adaptive_persistence.py` — `recall_artifacts_for_prompt()` filters by company_id
- `backend/agents/vectorstore.py` — `search_company_documents()` enforces `WHERE company_id = %s`
- `backend/agents/db.py` — `fetch_artifacts_by_owner()` scopes by owner_type + owner_id
- **GAP**: Chat route (`/conversations/<id>/messages`) does NOT re-check company membership after conversation creation
- **GAP**: No project-scoped RAG yet (Step 1.7 from prior plan)

Roles ⚠️

- Current: owner(4) / admin(3) / member(2) / viewer(1) in `backend/auth.py` ROLE_RANK
- **No "dev" role** — observability/miners/evidence are open to all authenticated users
- Admin frontend pages have zero role checks

LangChain / SQL Agent ❌

- No langchain, langgraph, or langchain_community in requirements.txt
- No per-company database connection concept
- No `company_databases` table
- Single shared PostgreSQL via psycopg2

Implementation Plan

Phase A: Miner Packaging & Developer Guide

Step A.1 — Create miner distribution archive
- Build script (`miner/build_dist.sh`) that runs `docker build -f Dockerfile.obfuscated -t elis-miner:latest .` then exports a tar.gz
- Include in archive: compiled .so files (no .py source), `start-with-ollama.sh` , `requirements.txt` , sample `.env.template` , `README.md`
- Alternative: push to private container registry and provide `docker pull` instructions

Step A.2 — Create `.env.template` for external miners
- Template with all required/optional vars: `BACKEND_URL` , `MINER_ENROLLMENT_SECRET` , `WORKER_ID` , `OPENAI_API_KEY` , `OPENAI_MODEL` , `MODEL_NAME` , `MINER_BACKEND` (openai|ollama|stub), `KBS_URL` ,

`KBS_ATTESTATION_SECRET`
- Each var gets a one-line comment explaining purpose and example value

Step A.3 — Write developer setup guide (`docs/miner-setup-guide.md`)
- Sections: Prerequisites, Download, Configuration, Docker Run, Manual Run, Registration Flow, Troubleshooting
- Include: `docker run` one-liner, `docker-compose` snippet for adding a miner, verification steps (check `/health` , confirm enrollment in backend)
- Reference the obfuscated Docker image download link (placeholder URL pattern for hosting)

Step A.4 — Add download endpoint or static hosting reference
- Option: serve the tar.gz from backend static route `GET /api/downloads/miner` (admin/dev-only)
- Option: document S3/Azure Blob/GCS hosting with signed URL generation
- Recommend: container registry pull (simplest for Docker users)

Phase B: Per-Company SQL Databases (LangChain SQL Agent)

Step B.1 — Add `company_databases` table to schema
- In `backend/agents/db.py` `init_schema()` : new table `company_databases(id, company_id FK, display_name, db_dialect, connection_string_encrypted, allowed_tables_json, read_only BOOLEAN DEFAULT true, created_by, created_at, updated_at)`
- `connection_string_encrypted` uses Fernet symmetric encryption (key from `DB_ENCRYPTION_KEY` env var)
- `allowed_tables_json` — explicit allowlist of tables the SQL agent can query (admin-configured)
- UNIQUE on `(company_id, display_name)`

Step B.2 — Add LangChain dependencies
- Add to `requirements.txt` : `langchain` , `langgraph` , `langchain-community` , `langchain-openai`
- Optional: `langsmith` for tracing (controlled by `LANGSMITH_TRACING` env var)

Step B.3 — Create `backend/agents/tools/sql_agent_tool.py`
- New tool that integrates with the existing tool system via `bundle_resolver.py`
- On invocation:
1. Receive `company_id` + `database_id` + `question` from tool args
2. Look up `company_databases` row, verify company membership, decrypt connection string
3. Create `SQLDatabase.from_uri(decrypted_uri, include_tables=allowed_tables)`
4. Create `SQLDatabaseToolkit(db=db, llm=model)` with read-only safeguards
5. Build LangChain SQL agent with system prompt enforcing: NO DML, list tables first, check query before executing, limit results
6. Run agent, return structured result with SQL query shown + results
- **Security**: connection string never exposed to LLM; `read_only=True` enforced at DB wrapper level;

DML blocked in system prompt AND via SQL parsing check
- Register as tool bundle "sql_query" in `bundle_resolver.py`

Step B.4 — Create CRUD API for company databases
- In `backend/company_api.py` or new `backend/company_db_api.py`:
- `POST /api/companies/<id>/databases` — admin/dev creates DB connection (encrypts connection string)
- `GET /api/companies/<id>/databases` — list configured databases (connection string masked)
- `PUT /api/companies/<id>/databases/<db_id>` — update display_name, allowed_tables, read_only
- `DELETE /api/companies/<id>/databases/<db_id>` — remove database config
- `POST /api/companies/<id>/databases/<db_id>/test` — test connection, return table list
- All routes require `@require_company_member(min_role="admin")` (or "dev" after role expansion)

Step B.5 — Integrate SQL tool into conversation flow
- When a conversation has `company_id`, check if company has databases configured
- Make `sql_query` tool available in that conversation's tool set
- User asks natural language question → orchestrator routes to SQL agent tool → LangChain agent generates/validates/executes SQL → results flow back through evidence pipeline
- SQL queries and results appear in trace (step_type="tool", name="sql_query")

Step B.6 — Frontend: Database management UI
- In company detail page (`/companies/[id]`), add "Databases" tab (visible to admin/dev roles)
- Form: display name, dialect dropdown (postgresql/mysql/sqlite/mssql), connection string (password field), allowed tables multiselect (populated from test-connection), read-only toggle
- Test connection button → shows discovered tables → user selects which to allow
- List of configured databases with edit/delete

Step B.7 — Human-in-the-loop SQL approval (optional, Phase 2)
- For destructive-looking queries (even if read-only is set), surface a confirmation step in chat
- Use LangChain HumanInTheLoopMiddleware pattern with `interrupt_on={"sql_db_query": True}` for "dangerous" detection
- This can be deferred; initial version trusts read-only + DML blocking

---

Phase C: Role Expansion (user/admin/dev)

Step C.1 — Expand role enum and ROLE_RANK
- In `backend/agents/db.py`: ALTER company_members role CHECK to include 'dev': `CHECK (role IN ('owner', 'admin', 'dev', 'member', 'viewer'))`
- In `backend/auth.py`: update `ROLE_RANK = {"owner": 5, "admin": 4, "dev": 3, "member": 2, "viewer": 1}`

- "dev" sits between admin and member — can access technical features but cannot manage members/billing

Step C.2 — Define role capabilities matrix
- **owner** (5): Everything. Transfer ownership.
- **admin** (4): Manage members, documents, agents, databases, billing, audit log. Cannot delete company.
- **dev** (3): Access miners page, observability/traces, evidence footer detail, database management, "How this answer was formed" detailed view. Can create/manage company agents.
- **member** (2): Chat, view documents, view project list. Basic evidence footer (summary only).
- **viewer** (1): Read-only access to shared conversations and documents.

Step C.3 — Add `require_company_role` feature-gating decorator
- New decorator or extend existing `require_company_member` to support role-specific feature checks
- Example: `@require_company_member(min_role="dev")` for observability/miner routes
- For frontend, add role check to company context: `GET /api/companies/<id>` returns `my_role` field (already does via membership query)

Step C.4 — Gate admin/observability API endpoints by role
- `backend/api.py`: Add company-context awareness to admin endpoints
- `/api/admin/runs/*`, `/api/admin/conversations/*` — scoped by company when `company_id` param present; require dev+ role
- Personal (no company) traces remain visible to the user who owns them
- `backend/miner_api.py`: `/api/miners` list and detail — require dev+ role for company-context miners; personal miners remain user-accessible

Step C.5 — Gate frontend features by role
- Evidence footer (`evidence-footer.tsx`): Full detail (SQL queries, step traces, tool inputs/outputs) only for dev+ role; member/viewer see summary only (source categories + trust badges, no raw data)
- Miners page: Show "Requires dev access" message for users below dev role within company context
- Admin/observability pages: Check company role in frontend, show gated UI or redirect
- Company detail page: Databases tab only visible to admin/dev
- Sidebar: "Observability" link requires dev+ company role

Step C.6 — Update invite flow for new role
- `backend/company_api.py` invite endpoint: allow `role="dev"` in invitation
- Frontend invite dialog: add "Developer" option in role dropdown
- Update role badges and display across member lists

Phase D: Validate Personal Data Redaction

Step D.1 — Write validation test script (`tests/test_redaction.py`)
- Test entities: name, email, phone, SSN, credit card, address, API key, DOB, bank account, passport, driver's license, password
- For each entity type:
1. Submit a prompt containing the PII via `/api/conversations/<id>/messages`
2. Inspect the run trace (`/api/admin/runs/<id>`) — verify the `step_traces` show redacted placeholders (`«PERSON_0»` , `«EMAIL_1»` , etc.) in miner-bound inputs
3. Verify the final user-facing output has rehydrated values (original PII restored)
4. Verify the stored trace does NOT contain raw PII in miner-visible fields

Step D.2 — Validate each privacy zone
- **Zone 1 (SAFE)**: City names, public company names → appear un-redacted in all contexts
- **Zone 2 (REDACT_FOR_MINER)**: Email, phone, address, IP → redacted before miner, rehydrated in output
- **Zone 3 (REDACT_ALWAYS)**: SSN, credit card, DOB → redacted everywhere, even in stored traces; verify they don't leak into step_traces or audit log

Step D.3 — Validate tool-specific privacy
- Test tool calls that require PII (e.g., `send_email` tool needs email in "to" field)
- Verify `TOOL_RESOLUTION_POLICY` in `privacy_policy.py` allows the specific entity type through for that field only
- Verify other fields still redact

Step D.4 — Validate env var controls
- Test with `REDACT_ENABLED=false` → no redaction occurs (baseline)
- Test with `REDACT_ENABLED=true, REDACT_NER_ENABLED=false` → only regex patterns caught
- Test with both enabled → full NER + regex detection

Phase E: Validate Company/Project Artifact Isolation

Step E.1 — Test: company docs don't appear in personal conversations
- Create company with documents (upload + wait for processing)
- Create a personal conversation (no company_id)
- Ask a question that matches company document content
- Verify: response does NOT cite company documents; evidence footer shows zero `retrieved_document` entries from company

Step E.2 — Test: company A docs don't appear in company B conversations
- Create two companies with different documents
- Create conversation in company A context

- Ask question matching company B's documents
- Verify: only company A documents retrieved

Step E.3 — Test: project-scoped RAG (requires implementing Step 1.7 from prior plan)
- **Pre-requisite**: Implement project-scoped document retrieval in `adaptive_persistence.py` / `vectorstore.py`
- Link document to project A but not project B (both in same company)
- Create conversation in project A → verify document is retrieved
- Create conversation in project B → verify document is NOT retrieved
- Create company-level conversation (no project) → verify company-wide documents retrieved

Step E.4 — Test: removed member can't access company artifacts
- Add user to company, create conversation
- Remove user from company
- Attempt to send message to existing conversation → should get 403 (after fix in Phase F)
- Attempt to create new conversation with company_id → should get 403

Phase F: Validate Company Agent Access Control

Step F.1 — Fix: re-check company membership on chat route
- In `backend/api.py` `send_message()`: after fetching conversation's company_id, re-verify membership
- If conversation has company_id and user is no longer an active member → return 403
- This closes the gap identified in exploration

Step F.2 — Test: non-member can't use company agents
- Create company agent with custom system prompt
- Create conversation with company_agent_id as a member → works
- Remove user from company → attempt to chat → 403

Step F.3 — Test: company agent system prompt doesn't leak to other companies
- Company A has agent with secret system prompt "You are FinanceBot"
- Company B conversation should never see "FinanceBot" in any trace
- Verify by inspecting `router_trace.company_agent_override` in runs for company B

Step F.4 — Test: project-level agent isolation (if applicable)
- Project member can access project resources
- Non-project member within same company cannot access project conversations (require project membership check — already implemented in `project_api.py` via `_require_project_member`)

Relevant Files

Backend — Modify

- `backend/agents/db.py` — Add `company_databases` table, expand role CHECK, add CRUD methods
- `backend/auth.py` — Update ROLE_RANK to include "dev" at rank 3 (shift admin→4, owner→5)
- `backend/company_api.py` — Update role validation to accept "dev", add database CRUD routes
- `backend/api.py` — Fix membership re-check on chat route; add role-gating to admin endpoints
- `backend/agents/adaptive_persistence.py` — Add project-scoped document filtering in `recall_artifacts_for_prompt()`
- `backend/agents/tools/bundle_resolver.py` — Register "sql_query" tool bundle

Backend — Create

- `backend/agents/tools/sql_agent_tool.py` — LangChain SQL agent tool integration
- `backend/company_db_api.py` — (optional, or add to company_api.py) Database config CRUD endpoints
- `docs/miner-setup-guide.md` — Developer guide for miner setup
- `miner/.env.template` — Template env file for external miners
- `miner/build_dist.sh` — Build script for obfuscated miner archive
- `tests/test_redaction.py` — PII redaction validation tests
- `tests/test_company_isolation.py` — Artifact/agent isolation validation tests

Frontend — Modify

- `frontend/components/blocks/evidence-footer.tsx` — Gate detailed view behind dev+ role
- `frontend/app/(app)/admin/*` — Add role checks to observability pages
- `frontend/app/(app)/miners/page.tsx` — Add role checks
- `frontend/app/(app)/companies/[id]/page.tsx` — Add "Databases" tab for admin/dev
- `frontend/components/layout/sidebar.tsx` — Conditionally show Observability link based on role

Config

- `requirements.txt` — Add langchain, langgraph, langchain-community, langchain-openai

- `.env` / Docker Compose — Add `DB_ENCRYPTION_KEY`, `LANGSMITH_TRACING`, `LANGSMITH_API_KEY`

Verification

1. **Miner packaging**: Build obfuscated image, extract tar.gz, run on fresh machine with only `.env` config → miner enrolls and receives jobs
2. **SQL agent**: Configure test PostgreSQL DB for a company → ask natural language question in chat → verify correct SQL generated, executed, results returned in evidence
3. **Role gating**: Log in as member (not dev) → confirm observability pages show "access denied", evidence footer shows summary only, miners page gated
4. **Redaction**: Run `tests/test_redaction.py` → all entity types properly redacted per zone, rehydrated in output
5. **Artifact isolation**: Run `tests/test_company_isolation.py` → company A docs never appear in company B or personal conversations; project scoping works
6. **Agent isolation**: Removed member gets 403 on existing company conversation; company agent prompt doesn't leak cross-company

---

Decisions

- **Role hierarchy**: owner(5) > admin(4) > dev(3) > member(2) > viewer(1). "dev" is a new role between admin and member.
- **Dev capabilities**: miners, observability/traces, evidence detail, database management, company agents. Cannot manage members or billing.
- **SQL agent approach**: LangChain `SQLDatabaseToolkit` wrapped as a custom tool in the existing tool system, not replacing the orchestrator.
- **Database connections**: Encrypted at rest (Fernet), admin/dev-configured per company, read-only by default, explicit table allowlist.
- **Miner distribution**: Obfuscated Docker image (Cython-compiled) via container registry pull; tar.gz archive as alternative.
- **Human-in-the-loop SQL**: Deferred to Phase 2; initial version relies on read-only + DML blocking.
- **Out of scope**: Direct database write operations, cross-company database sharing, SQL agent for personal (non-company) conversations, LangSmith integration (optional add-on).

Further Considerations

1. **Document-level vs project-level access**: Current plan uses project membership as the access boundary. Should individual documents within a project have their own ACL? *Recommendation: Start with project-level access (simpler), add document-level ACL later if needed.*
2. **Company-wide documents**: Should there be a concept of "company-wide" documents visible to all company members regardless of project? *Recommendation: Yes — documents not linked to any project are company-wide by default, accessible to all company members.*
3. **Sidebar UX**: The sidebar currently shows a flat conversation list. Switching to company→project→conversation hierarchy is a significant UX change. *Recommendation: Add a collapsible org/project tree above the existing conversation list, with "Personal" as the default view.*

Repository: Analytics

This is a **great benchmark idea** because it tests several hard things simultaneously:

• tool selection
• multi-step retrieval
• long-context memory
• cross-turn reasoning
• aggregation across sources
• projections using external data

I'll structure this so **each industry becomes one 10-turn conversation**:

1 initial analytics question

- 9 follow-ups that **require web retrieval** and **reuse earlier facts**

Your LLM must:

- search
- extract numbers
- remember earlier results
- combine them later

If it tries to answer from training data, it will fail.

Industry Internet-Analytics Benchmark Conversations

Each industry = **1 long conversation (10 turns)**

---

1. Finance / Stock Market

Turn 1

What are the **current market caps and P/E ratios** of Apple, Microsoft, Nvidia, Amazon, and Google?

Turn 2

Calculate the **average P/E ratio of those five companies**.

Turn 3

Which of those companies has had the **largest stock price increase over the past 12 months**?

Turn 4

What was the **revenue growth rate** for that company in the most recent earnings report?

Turn 5

Compare the **R&D spending** of Apple vs Microsoft last fiscal year.

Turn 6

Using the numbers above, estimate the **R&D as a percentage of revenue** for each.

Turn 7

What are analysts currently forecasting as the **AI market growth rate through 2030**?

Turn 8

Using those forecasts, estimate how much the **AI revenue exposure** of Nvidia could grow.

Turn 9

Find the **latest interest rate decision by the Federal Reserve**.

Turn 10

Based on the rate decision and current valuations, which of these companies would likely be **most sensitive to higher rates**?

---

2. Healthcare / Pharma

Turn 1

What are the **top 5 pharmaceutical companies by revenue in 2024**?

Turn 2

What were the **total revenues** for each of them last year?

Turn 3

Which of those companies currently has the **largest oncology drug portfolio**?

Turn 4

List the **top selling oncology drugs globally and their revenue**.

Turn 5

Which companies manufacture those drugs?

Turn 6

Calculate the **estimated oncology market share** for each company.

Turn 7

What is the **global oncology drug market growth projection through 2030**?

Turn 8

What new **cancer drugs received FDA approval in the last 12 months**?

Turn 9

Which of those approvals are expected to generate **blockbuster revenue (> $1B annually)**?

Turn 10

Based on this data, which pharma company is **best positioned to dominate oncology in the next decade**?

---

3. Energy / Oil & Gas

Turn 1

What are the **largest oil companies by production volume today**?

Turn 2

What was their **daily oil production in barrels per day**?

Turn 3

Which of those companies has the **largest renewable energy investments**?

Turn 4

What is the **global oil demand forecast for 2030**?

Turn 5

What is the **projected growth rate of solar energy worldwide**?

Turn 6

Which oil companies are investing the **most into solar or wind projects**?

Turn 7

Find the **latest crude oil price per barrel**.

Turn 8

What geopolitical events recently affected oil prices?

Turn 9

Which oil producing countries currently control the **largest reserves**?

Turn 10

Based on reserves, renewables investment, and demand forecasts, which energy company is **best positioned for the next 20 years**?

---

4. Technology / AI

Turn 1

What companies currently lead the **AI chip market**?

Turn 2

What is the **market share of Nvidia, AMD, Intel, and others**?

Turn 3

What is the **projected size of the AI hardware market by 2030**?

Turn 4

Which companies are building the **largest AI data centers**?

Turn 5

What is the **estimated cost of building a hyperscale AI data center**?

Turn 6

How much electricity does a large AI data center consume?

Turn 7

Which countries currently host the **largest number of AI data centers**?

Turn 8

What is the **global semiconductor shortage outlook**?

Turn 9

Which companies are building **new semiconductor fabs**?

Turn 10

Based on these trends, which company has the **largest long-term AI infrastructure advantage**?

---

5. Retail / Ecommerce

Turn 1

What are the **largest ecommerce companies globally by revenue**?

Turn 2

What is the **current global ecommerce market size**?

Turn 3

What percentage of ecommerce sales are **mobile vs desktop**?

Turn 4

Which regions have the **fastest ecommerce growth**?

Turn 5

What are the **average ecommerce conversion rates by industry**?

Turn 6

What logistics companies support the **largest ecommerce platforms**?

Turn 7

What is the **average shipping cost per order** globally?

Turn 8

What percentage of ecommerce sales now happen through **marketplaces vs independent stores**?

Turn 9

Which companies dominate **last-mile delivery**?

Turn 10

Based on these numbers, which ecommerce company has the **most defensible logistics moat**?

---

6. Automotive / EV

Turn 1

What are the **largest electric vehicle manufacturers by sales**?

Turn 2

How many EVs did Tesla, BYD, and Volkswagen sell last year?

Turn 3

What is the **global EV market share** of those companies?

Turn 4

What is the **average battery cost per kWh today**?

Turn 5

Which companies manufacture the **largest EV battery supply**?

Turn 6

What are the **largest lithium mines globally**?

Turn 7

What is the **global EV adoption forecast through 2035**?

Turn 8

Which countries have the **highest EV incentives**?

Turn 9

Which automakers are investing the **most in EV factories**?

Turn 10

Based on production capacity and battery supply, which automaker will likely **lead the EV market in 2030**?

---

7. Real Estate

Turn 1

What are the **current average home prices in the top 10 US cities**?

Turn 2

What are the **current mortgage interest rates**?

Turn 3

How have home prices changed **year over year in those cities**?

Turn 4

Which cities currently have the **highest rent growth**?

Turn 5

What are the **vacancy rates for commercial real estate**?

Turn 6

What sectors of real estate are growing fastest (industrial, residential, office)?

Turn 7

What companies are the **largest real estate investment trusts (REITs)**?

Turn 8

What is the **current construction cost per square foot** in major cities?

Turn 9

What are demographic projections for **urban population growth**?

Turn 10

Which real estate markets appear **most undervalued today**?

---

8. Aviation

Turn 1

What are the **largest airlines by passenger volume globally**?

Turn 2

What is the **average airline profit margin**?

Turn 3

Which aircraft manufacturers dominate the **commercial jet market**?

Turn 4

What are the **production backlogs for Boeing and Airbus**?

Turn 5

What is the **average cost of a Boeing 737 or Airbus A320**?

Turn 6

What are the **largest airports by passenger traffic**?

Turn 7

What are the **global air travel growth forecasts**?

Turn 8

What are the **largest airline alliances**?

Turn 9

How have **jet fuel prices changed recently**?

Turn 10

Which airlines are best positioned for **long-term growth**?

---

9. Agriculture / Food

Turn 1

What are the **largest agricultural producers globally by crop output**?

Turn 2

What are the **current global wheat, corn, and soybean prices**?

Turn 3

Which countries export the **most grain**?

Turn 4

What is the **global fertilizer market size**?

Turn 5

Which companies produce the **most agricultural chemicals**?

Turn 6

How has **climate change affected crop yields**?

Turn 7

What are the **largest food companies by revenue**?

Turn 8

What percentage of global food production is **wasted each year**?

Turn 9

What are the **largest agricultural technology companies**?

Turn 10

Based on production and climate risks, which countries will **dominate food exports in 2050**?

---

10. Media / Entertainment

Turn 1

What are the **largest streaming platforms by subscriber count**?

Turn 2

What are the **current subscriber numbers for Netflix, Disney+, Amazon Prime, and Max**?

Turn 3

What is the **average monthly streaming subscription price**?

Turn 4

Which companies produce the **most original content**?

Turn 5

What are the **highest grossing movies of the last 12 months**?

Turn 6

Which studios produced those films?

Turn 7

What is the **global box office revenue this year**?

Turn 8

How much do major studios spend **annually on content production**?

Turn 9

What percentage of media consumption is now **streaming vs traditional TV**?

Turn 10

Which streaming company has the **strongest long-term competitive advantage**?

What This Tests in Your System

These conversations force your system to:

1️⃣ Trigger web retrieval repeatedly

Every turn requires **fresh data**

2️⃣ Maintain memory

Later questions reuse earlier numbers.

3️⃣ Do math aggregation

Examples:

- averages
- percentages
- projections

4️⃣ Combine multiple sources

5️⃣ Resist hallucination

Most answers require **exact numbers**.

Even Better Stress Test

Run **all 10 industries simultaneously** as parallel sessions.

This tests:

- memory isolation
- session context handling
- retrieval concurrency
- tool usage reliability

If you'd like, I can also generate a **much harder benchmark (100-turn conversations)** that stress tests:

- tool routing
- reasoning chains
- multi-document synthesis
- recursive queries

Those are **extremely good at breaking LLM agent systems**.

Repository: Implementation Summary

BYOK API Key Failure Handling System - Implementation Summary

🎯 Objective Completed

Implemented a comprehensive system to **catch API key failures, automatically deactivate affected miners, create audit trails, and enable user recovery** for the BYOK (Bring Your Own Key) miner system.

✅ What Was Built

1. Database Infrastructure

- ✅ Migration `0088_byok_key_audit_log.py`: New audit trail table with 3 indexes
- ✅ `byok_key_audit_log` table: Tracks all key lifecycle events (auto-deactivation, manual deactivation, recovery)
- ✅ Enhanced `user_miner_api_keys`: Added `is_active`, `key_error`, `api_key_fingerprint` columns

2. Automatic Error Detection & Response

- ✅ Integration with existing dispatch error handling (backend/miner_scheduler.py)
- ✅ Detects: 401/403 HTTP errors, "unauthorized", "missing_api_key", "invalid_api_key"
- ✅ Raises: `_InternalAuthError` with provider name
- ✅ Triggers: `deactivate_byok_api_key_fingerprint()` automatically

3. Key Deactivation Engine

- ✅ Backend/services/platform_api_keys.py: Enhanced deactivation function with:
- Pre-update affected user query
- Audit log creation with metadata
- Notification emission to affected users
- Error reason tracking in key_error field

4. Audit Trail System

- ✅ Backend/agents/db.py: Three new methods

- `create_byok_audit_log()`: Log any lifecycle event
- `get_byok_audit_by_fingerprint()`: Query by key
- `get_byok_audit_by_user()`: Query by user
- ✅ Stores: action, reason, triggered_by, actor_type, affected_miners, provider, metadata

5. User Management Endpoints

- ✅ Backend/user_miner_byok_api.py: Three new REST endpoints
- `GET /api/miners/byok/keys/audit-log`: View key audit history
- `POST /api/miners/byok/keys/{fingerprint}/deactivate`: Manual deactivation
- `POST /api/miners/byok/keys/{fingerprint}/reactivate`: Recovery after fix
- ✅ All require authentication & verify user ownership

6. User Notifications

- ✅ Kind values: `"byok_key_deactivated"`, `"byok_key_recovered"`
- ✅ Metadata: provider, reason, miner_count, action_url
- ✅ Automatic emission on deactivation
- ✅ Automatic emission on recovery
- ✅ User sees in notification panel with ability to take action

📊 Test Results

All 5 comprehensive tests **PASSING**:

```
✓ Test 1: Audit log creation
✓ Test 2: Audit log queries (by fingerprint, by user)
✓ Test 3: User notifications integration
✓ Test 4: BYOK key deactivation workflow
✓ Test 5: Miner recovery after key fix

Results: 5 passed, 0 failed
```

🔄 Complete Workflows

Automatic Deactivation on Auth Error

```
User Request → Miner Dispatch → 401/403 Error
  ↓
_InternalAuthError raised
  ↓
Dispatcher catches & queries fingerprint
  ↓
deactivate_byok_api_key_fingerprint() called
  ↓
System performs:
  1. Sets is_active=false on all miners with fingerprint
  2. Stores error in key_error field (max 200 chars)
  3. Creates audit log entry (action="auto_deactivated")
  4. Queries affected users BEFORE update
  5. Emits notification to each user:
     - kind: "byok_key_deactivated"
     - title: "API Key Deactivated (provider)"
     - body: "{N} miners now offline. Please update key."
     - metadata: {fingerprint, provider, reason, miner_count}
  ↓
User sees notification → visits /miners → updates key
  ↓
User calls POST /reactivate
  ↓
System:
  1. Sets is_active=true, key_error=''
  2. Updates miners: status=online, disabled_reason=NULL
  3. Creates audit log (action="recovered")
  4. Emits "recovered" notification
  ↓
✅ Miners back online
```

Manual Key Deactivation

```
User suspects compromise
  ↓
User calls POST /api/miners/byok/keys/{fingerprint}/deactivate
  ↓
System verifies ownership
  ↓
Creates audit log (action="manual_deactivate", actor_type="user")
  ↓
Calls deactivate_byok_api_key_fingerprint()
  ↓
✅ All miners with key offline + notification sent
```

📁 Files Modified/Created

New Files

- ✅ `backend/migrations/0088_byok_key_audit_log.py` (43 lines)
- ✅ `_test_byok_key_failure_handling.py` (360+ lines)
- ✅ `BYOK_KEY_FAILURE_HANDLING.md` (comprehensive documentation)

Modified Files

- ✅ `backend/agents/db.py`: Added 3 audit log methods (~75 lines)
- ✅ `backend/services/platform_api_keys.py`: Enhanced deactivation (~100 lines added)
- ✅ `backend/user_miner_byok_api.py`: Added 3 endpoints (~200 lines added)

Total Implementation: ~800 lines of production code + comprehensive tests

---

🔐 Security Features

1. **Fingerprint-based Isolation**: All miners with same key deactivated together
2. **No Key Exposure**: Only fingerprints and error codes stored, never keys
3. **Full Audit Trail**: Every action logged with actor_type, reason, timestamp
4. **User Notifications**: Immediate alert when key fails
5. **Ownership Verification**: All endpoints verify user owns the key
6. **Graceful Degradation**: Notification failures don't block deactivation

---

🚀 Integration Status

- ✅ Backend compiles without errors
- ✅ All migrations applied successfully
- ✅ Database tables created and indexed
- ✅ All code integrated with existing systems
- ✅ Notification system ready (uses existing table)
- ✅ All tests passing

📋 Usage Examples

Check Key Status History

```
GET /api/miners/byok/keys/audit-log?limit=50
→ Returns audit log with all key events (auto-deactivations, manual actions, recoveries)
```

Deactivate Key (Manual)

```
POST /api/miners/byok/keys/{fingerprint}/deactivate
Body: {"reason": "Key compromise suspected"}
→ All miners with fingerprint go offline + notification sent
```

Recover Key After Fix

```
POST /api/miners/byok/keys/{fingerprint}/reactivate
Body: {"reason": "Key rotated and updated"}
→ All miners with fingerprint come back online + notification sent
```

📚 Documentation

Comprehensive documentation available in:
- BYOK_KEY_FAILURE_HANDLING.md: Full architecture, workflows, and API reference
- Technical summary for future reference

🔮 Future Enhancements

1. **Bulk Recovery**: Reactivate multiple keys at once
2. **Admin Dashboard**: System admin view of all key failures
3. **Compliance Reporting**: Generate audit reports
4. **Key Rotation Automation**: Support automated rotation
5. **Alert Escalation**: Escalate after N failures
6. **Proactive Validation**: Periodic test calls to detect issues early

✨ Impact

This system provides:
- **User Protection**: Immediate notification of key failures
- **System Reliability**: Automatic removal of broken miners from dispatch
- **Compliance**: Full audit trail of all key lifecycle events
- **Operability**: Clear recovery path for key issues
- **Transparency**: Users can view why their miners were disabled

---

Status: ✅ **COMPLETE AND TESTED**

Ready for production deployment.

Site: /docs

Documentation

Learn how the Elis AI distributed intelligence network processes your requests.

How Elis AI Works

Elis AI is a distributed intelligence network that breaks complex questions into smaller, focused tasks and routes them to specialized AI agents running across a decentralized pool of compute miners. Instead of relying on a single large model, every request passes through a dynamic pipeline that picks the best-fit agent, dispatches inference to the best available miner, and returns a fully traced answer — so you always know how a response was produced.

🤖 Elis AI Mode

Automatically builds and reuses specialized agents. Each agent runs a multi-step workflow — expertise lookup, context retrieval, execution — using the best available model tier for each step.

🔓 Elis Unlocked

Advanced multi-pass workflow with planning, fact-checking, and iterative refinement. Best for complex tasks that need verification and deep research.

⛏️ Miners

Miners host local LLMs and earn tokens for verified responses. The network supports multiple model tiers for speed and quality tradeoffs.

🛡️ Verification

Every inference step is traced end-to-end. The scheduler selects the best available miner using weighted round-robin, trust tiers, and load awareness — ensuring fair distribution and reliable execution.

Why the network stays sustainable

Elis is built on a token economy where compute contributors are rewarded for useful work. Miners that return high-quality results quickly earn more assignments. Because miners compete regionally, users get served by nearby high-performing nodes — improving latency and experience.

For Consumers

Contribute background compute to earn tokens that offset usage costs.

Background contribution earns tokens.

Tokens are spent on model and tool usage.

Quality verification keeps rewards fair.

For Enterprise

End-to-end encryption, automated PII redaction, audit trails, and self-hosted deployment options. See Enterprise section →

Process Flow

1. User submits prompt → 2. Router chain selects agent → 3. Agent builds workflow → 4. Miners run inference → 5. Quality verification → 6. Traced response returned

Ready to try it?

Create an account and start asking questions — every answer comes with a full trace.

Get Started → View Whitepapers

Site: /tutorials

Guided Walkthroughs
Tutorials shaped like the landing page, but built for real product work.
Step-by-step guides to help you get the most out of Elis AI. Choose the track that matches how you use the platform.
User Guide
For everyone. Learn how to use the platform day to day, set up teams, and manage the resources tied to your workspace.
Sign Up & Log In
Create your account and access the platform.
Open
Starting a Conversation
Send your first prompt and see AI responses stream in.
Open
Document Editor
Writer documents (50 cap), AI rewrite & selection tools, links, merge, history, and side chat.
Open
Managing Messages
Edit, resend, copy, and delete messages.
Open
Creating an Organization
Set up an organization to collaborate with your team.
Open
Managing Members
Invite team members, set roles, and manage access.
Open
Uploading Documents
Upload files for your organization's knowledge base.
Open
Connecting Databases
Link an external database for context-aware queries.
Open
Creating & Managing Projects
Organize work into projects with linked resources.
Open
Viewing the Audit Log
Review all activity within your organization.
Open

Settings & API Keys
Manage your profile and create API keys.
Open
Billing & Tokens
Purchase token packages and view transaction history.
Open
Subscription Management
Cancel or restore personal and organization subscriptions before access changes take effect.
Open
Organization Seat Management
Lower seat blocks safely by choosing which members or pending invites lose access.
Open
Browsing Models
Explore the model directory and compare options.
Open
Accepting Invitations
View and respond to pending organization invites.
Open
Project Conversations
Create and manage conversations within a project context.
Open
Using Dashboards
Create dashboards, iterate drafts, publish versions, and manage favorites.
Open
AI Edit a Dashboard
Edit dashboards in plain English with a live preview, clarify questions, and per-message restore points.
Open
Dashboard Editor Deep Dive
Understand each editor section: planning, AI spec edits, tasks, buckets, datasources, draft thread, and assistants.
Open
Using Automations
Organization-only recurring widgets with mobile support, chart modes, and alert dispatch.
Open
Deep Data Collection
Crawl websites, extract structured records, share public datasets, and earn credits.
Open
Enterprise & Dedicated Hosting

Enterprise
Dedicated infrastructure, model governance, BYOK/BYOA credentials, organization-scoped miners, and lifecycle operations for enterprise deployments.
Dedicated Hosting Upgrade
Activate dedicated deployment, select shared vs. dedicated at org creation, and monitor cluster provisioning lifecycle.
Open
Dedicated User Management
Invite members with cluster-aware links, review pending signup invites, and adjust paid seat capacity without losing track of reserved seats.
Open
Enterprise Controls
Configure model allow/block lists, web domain allowlist, BYOK API keys, and BYOA OAuth credentials.
Open
Provider API Keys (BYOK)
Sign up with OpenAI, Anthropic, Google, Groq, Mistral, and Cerebras, then store each provider key on your organization.
Open
Dedicated Miners
Assign organization-scoped miners, generate registration tokens, and enforce dedicated-only inference.
Open
OAuth & SSO Setup
Configure BYOA OAuth credentials and optional SSO enforcement for enterprise cluster members.
Open
Dedicated Lifecycle Operations
Operate Stripe webhook lifecycle, grace mode, data export, and cluster decommission procedures.
Open
Developer Guide
For admins and developers. Go deeper on miner management, API integration, observability, and operator tooling.
On-Prem Installation
Purchase the annual license, download the Docker bundle, install dependencies, and activate the platform.
Open
Setting Up a Miner
Hosted or self-hosted: BACKEND_URL, MODEL_NAME, pairing key, Docker or Windows.
Open

Managing Miners

Monitor health, filter workers, enable/disable, validate pairing, unclaim when needed.

Open

Using the API

Authenticate and make programmatic requests.

Open

Automations API

Create, run, and monitor recurring research widgets via REST.

Open

Conversation Explorer

Browse and filter conversations with run telemetry.

Open

Agent Directory & Detail

Search agents, view stats, and explore mutations.

Open

Runs Dashboard & Comparison

Filter runs and compare them side by side.

Open

Run Detail Deep Dive

Inspect Overview, Routing, Steps, Context, and Integrity tabs.

Open

Anomaly Dashboard

Filter by anomaly type and severity, then drill into flagged runs.

Open

Downloads

Access compiled packages and Docker images.

Open