SafeClaw — Offline-First, RAG-First, MCP-Exposed Stack

Production-grade Python system for `.md`-corpus RAG with:

- **LangGraph controller** enforcing RAG-first via graph topology (not prompts)
- **FastAPI gateway** with user confirmation flow for gated Grok fallback
- **Hybrid retrieval** (ChromaDB semantic + BM25 keyword) with RRF fusion
- **MCP server** (retrieval-only, no sampling capability)
- **sentence-transformers** for CPU-only local embeddings (no Ollama)

Architecture

Invariants (Enforced by Code, Not Prompts)

1. Every query passes through retrieval first
2. No LLM is called before the score gate
3. No Grok without explicit user confirmation AND hybrid mode
4. Every response passes through audit logging

Quick Start

1. Install

```bash
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt
```

2. Index Corpus

Place `.md` or `.txt` files in `data/corpus/`, then:

```bash
python -m retrieval.indexer
```

This builds both ChromaDB (semantic) and BM25 (keyword) indices using sentence-transformers `all-MiniLM-L6-v2` on CPU.

3. Start LM Studio

Load a GGUF model (e.g., Qwen 2.5 7B Instruct) in LM Studio. Ensure the server is running on `http://127.0.0.1:1234`.

4. Run Gateway

```bash
python gate.py
```

Gateway binds to `127.0.0.1:8787` (localhost only).

5. Query

```bash
```

```
# High-confidence query (local LLM answers directly)
curl -X POST http://127.0.0.1:8787/query \
  -H "Content-Type: application/json" \
  -d '{"query": "What is Veeam immutability?"}'

# Low-confidence query (triggers confirmation flow)
curl -X POST http://127.0.0.1:8787/query \
  -H "Content-Type: application/json" \
  -d '{"query": "Explain quantum physics basics"}'
# Response: {"needs_confirm": true, "confirm_message": "Vault miss..."}

# Re-submit with confirmation
curl -X POST http://127.0.0.1:8787/query \
  -H "Content-Type: application/json" \
  -d '{"query": "Explain quantum physics basics", "user_confirmed_online": false}'
```

Hybrid Mode (Grok Fallback)

To enable Grok fallback:

1. Set `app.mode: "hybrid"` and `models.grok.enabled: true` in `config.yaml`
2. Export your API key: `export GROK_API_KEY=your_key`
3. Restart the gateway

Grok is only called when ALL conditions are met:

- `app.mode == "hybrid"`
- `models.grok.enabled == true`
- `GROK_API_KEY` is set
- User explicitly confirms online escalation

MCP Server (Retrieval Only)

```
bash

# stdio transport
echo '{"jsonrpc":"2.0","id":1,"method":"initialize","params":{"protocolVersion":"2025-11-25"}}' \
  | python mcp_hybrid_server.py
```

The MCP server exposes `hybrid_search` only. `sampling: null` is set at protocol level — the MCP server **cannot** invoke an LLM.

Testing

```bash
# Run all tests (mocked — no live services required)
pytest tests/ -v

# Run specific test categories
pytest tests/test_stemmer.py -v          # Stemmer unit tests
pytest tests/test_sanitizer.py -v        # Prompt filter tests
pytest tests/test_audit.py -v            # Audit logging tests
pytest tests/test_graph.py -v            # LangGraph path tests
pytest tests/test_gate.py -v             # FastAPI endpoint tests
pytest tests/test_hybrid_search.py -v    # RRF fusion math tests
```

Metrics

```bash
python metrics.py
```

Parses `logs/audit.jsonl` and reports hit rate, score distribution, model usage, and query volume.

Configuration

All behavior controlled via `config.yaml`. Key settings:

Setting	Description	Default
`app.mode`	`offline` or `hybrid`	`offline`
`retrieval.min_score`	Score threshold for local_llm path	`0.75`
`models.grok.enabled`	Enable Grok fallback	`false`
`policy.prompt_filter.enabled`	Input sanitization	`true`
`api.host`	Gateway bind address	`127.0.0.1`
`api.port`	Gateway port	`8787`

Security

- **Localhost only**: Gateway and LM Studio bind to `127.0.0.1`
- **No Ollama**: Embeddings are local sentence-transformers (no extra server)
- **Prompt filter**: Banned patterns stripped from input and corpus
- **Privacy redaction**: Emails, IPs, secrets redacted from audit logs
- **Query hashing**: Audit log stores SHA256 hashes, not raw queries
- **MCP sampling disabled**: Protocol-level guarantee of no LLM in MCP
- **No third-party tools**: Only `hybrid_search` exposed, hardcoded

Project Structure

```
safeclaw/
├── gate.py                 # FastAPI gateway (HTTP entry point)
├── graph.py                # LangGraph state machine (controller)
├── mcp_hybrid_server.py    # MCP server (retrieval-only, stdio)
├── config.yaml             # Controller-grade configuration
├── requirements.txt
├── metrics.py              # Audit log analysis
├── retrieval/
│   ├── embeddings.py       # sentence-transformers wrapper (CPU)
│   ├── hybrid_search.py    # ChromaDB + BM25 + RRF fusion
│   ├── indexer.py          # Corpus ingestion + index builder
│   └── stemmer.py          # Enhanced Porter stemmer
├── llm/
│   └── client.py           # LM Studio + Grok clients
├── schemas/
│   └── api.py              # Pydantic request/response models
├── utils/
│   ├── errors.py           # Typed exception hierarchy
│   ├── health.py           # Dependency health checks
│   ├── logger.py           # Audit logging (JSONL + hashing)
│   └── sanitizer.py        # Prompt injection filter
├── data/corpus/            # Your .md/.txt files
├── index/                  # ChromaDB + BM25 indices
├── logs/                   # Audit and application logs
└── tests/
    ├── conftest.py         # Shared mocks and fixtures
    ├── test_stemmer.py
    ├── test_sanitizer.py
    ├── test_audit.py
```

```
├── test_hybrid_search.py
├── test_graph.py          # LangGraph integration tests
└── test_gate.py           # FastAPI endpoint tests
```