Axiomeer Product Guide

The Marketplace for AI Agents — v5 Prototype

Axiomeer

February 2026

Contents

Prototype Notice

Axiomeer is an early-stage prototype. The system works end-to-end, but the UX, feature set, and APIs will change quickly. This guide explains how it works today and how to use it.

Executive Summary

Think of Axiomeer as an **App Store for AI**.

Anyone can publish a product — a dataset, an API, a bot, or a model endpoint. Any AI agent can shop the marketplace, pick the best product for a job, execute it, validate the result, and ingest it through a single standard protocol.

This is not another tool-calling framework. It is infrastructure for an **AI-to-AI economy** where agents autonomously find and consume the right resources, and every transaction is verified.

Axiomeer is built for reliability. If the system cannot find a strong match, it returns `NO_MATCH` instead of guessing. Every execution is validated for citations and logged as a receipt, so results are traceable and auditable.

Product Overview

What Axiomeer Does

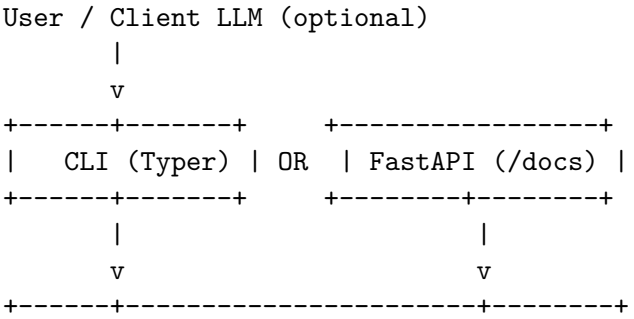
Axiomeer helps an agent:

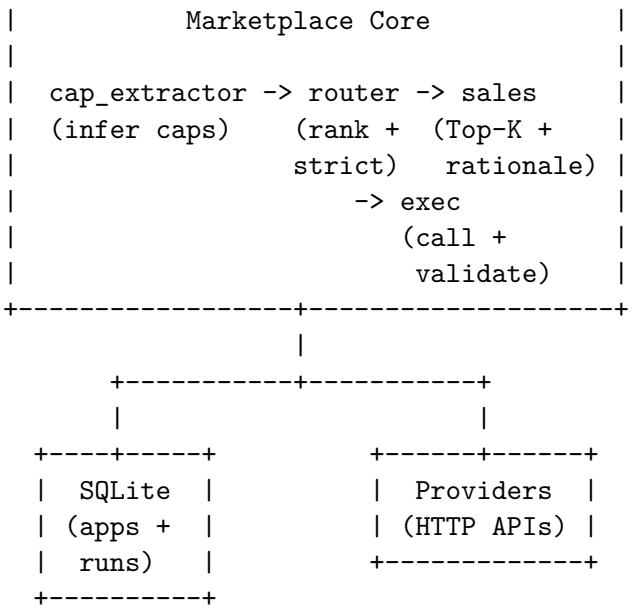
- Discover tools for a task
- Infer capability tags from the request (or use manual tags)
- Rank apps using scores and strict thresholds
- Recommend the best option with a short rationale
- Execute the chosen provider
- Validate citations and timestamps
- Record a run receipt in SQLite

Why It Matters

Wrong tool choices lead to wrong answers. Axiomeer tries to prevent that by being strict about routing and validation.

Architecture Overview





Flow

1. Client sends a task
2. Capabilities are inferred or provided manually
3. Router ranks apps with strict thresholds
4. Sales agent summarizes top-K options
5. Top app is executed via its `executor_url`
6. Output is validated
7. A receipt is logged to SQLite

Getting Started

System Requirements

- Python 3.10+
- Ollama (local LLM runtime)
  - Required model: `phi3.5:3.8b`

Install

```
git clone https://github.com/ujjwalredd/axiomeer.git
cd axiomeer

python3 -m venv .venv
source .venv/bin/activate

python -m pip install -e ".[dev]"

ollama pull phi3.5:3.8b
```

Run the API

```
uvicorn apps.api.main:app --reload
```

The server runs at `http://127.0.0.1:8000` with docs at `/docs`.

Core Concepts

Capabilities

Capabilities are tags such as `weather`, `finance`, `docs`, `citations`. You can let the LLM infer them with `--auto-caps` or specify them yourself with `--caps`.

Constraints

Constraints prevent unsafe tool matches.

- `citations_required`
- `freshness` (static, daily, realtime)
- `max_latency_ms`
- `max_cost_usd`

Recommendation Engine

The router uses weighted scoring. The values are normalized at runtime.

Weight	Factor	Description
0.70	Capability Match	Coverage ratio of required vs. available capabilities
0.20	Latency	Penalizes high latency with a soft curve
0.10	Cost	Prefers cheaper without over-penalizing
0.15	Trust Score	Success rate, citation pass rate, and latency percentile
0.25	Relevance	TF-IDF similarity between task and app metadata

Strict thresholds

- `MIN_CAP_COVERAGE` (default 1.0)
- `MIN_RELEVANCE_SCORE` (default 0.08)
- `MIN_TOTAL_SCORE` (default 0.55)

Validation and Auditing

- Every execution returns `ok` plus `validation_errors` if citations or timestamps are missing.
- Every run is logged to SQLite with latency, timestamps, and output.

End-to-End Examples

Real Query (Should Succeed)

```
python -m marketplace.cli shop \
  "What is the currency rate of usd to eur? Cite sources." \
  --auto-caps --citations --execute-top
```

Expected output (abbreviated)

```
Auto capabilities: ['finance', 'citations']
Status: OK
Determine the currency rate of USD to EUR.
Top Recommendations: exchange_rates
Execution result: ok=True
Output: Exchange rates for 1 USD: EUR: 0.84687 ...
```

Fake Query (Should Refuse)

```
python -m marketplace.cli shop \
  "Translate this sentence to Spanish: hello world. Cite sources." \
  --auto-caps --citations --execute-top
```

Expected output

```
Auto capabilities: ['translate', 'citations']
Status: NO_MATCH
No apps met minimum capability coverage (1.00).
No recommendations.
```

Model Comparison (Sales Agent)

Current small model: `phi3.5:3.8b`

Previous large model: `qwen2.5:14b-instruct`

Benchmarks below were collected on CPU. With GPU infrastructure, larger models can be used and can improve quality and speed depending on hardware.

Query	Small Model (s)	Large Model (s)	Delta (Large - Small)
Weather Indianapolis	18.15	22.70	+4.55
USD to EUR	19.19	44.98	+25.79
What is Python	21.80	46.92	+25.12
France population	18.79	48.28	+29.49
Define serendipity	19.13	27.22	+8.09
Books about Python	19.36	47.46	+28.10
Wikidata entity	20.93	48.71	+27.78
Wikipedia dump	17.56	27.91	+10.35

Publishing a New App

1. Create a manifest JSON:

```
{
  "id": "realtime_weather_agent",
  "name": "Realtime Weather Agent v2",
  "description": "Returns current weather with sources and timestamps.",
  "capabilities": ["weather", "realtime", "citations"],
  "freshness": "realtime",
  "citations_supported": true,
  "latency_est_ms": 800,
  "cost_est_usd": 0.0,
  "executor_type": "http_api",
  "executor_url": "http://127.0.0.1:8000/providers/openmeteo_weather"
}
```

2. Publish:

```
python -m marketplace.cli publish manifests/realtime_weather_agent.json
```

3. Verify:

```
python -m marketplace.cli apps
```

Configuration

Copy `.env.example` to `.env` and edit.

Key settings

- `SALES_AGENT_MODEL=phi3.5:3.8b`
- `ROUTER_MODEL=phi3.5:3.8b`
- `MIN_CAP_COVERAGE=1.0`
- `MIN_RELEVANCE_SCORE=0.08`
- `MIN_TOTAL_SCORE=0.55`
- `SALES_AGENT_TOP_K=8`

FAQ

Q: Why do I see `NO_MATCH`?

A: Routing is strict by design. If capability coverage or relevance is weak, Axiomeer refuses to guess.

Q: Can I connect RAG, datasets, or web search?

A: Yes. Expose an HTTP endpoint that returns JSON with citations and publish a manifest.

Q: How do I add a new provider?

A: Create a manifest, implement the endpoint, then publish via `python -m marketplace.cli publish`.

Q: Where are run receipts stored?

A: In SQLite (`marketplace.db`) under the `runs` table.

Q: How do I get a `run_id`?

A: Use `python -m marketplace.cli runs --n 10` or call `GET /runs` in the API.

Q: How do I fetch a full receipt?

A: `python -m marketplace.cli run <run_id>` or `GET /runs/{run_id}`.

Q: Can I skip the LLM and use manual capabilities?

A: Yes. Pass `--caps` and avoid `--auto-caps`.

Q: How do I change the sales-agent model?

A: Set `SALES_AGENT_MODEL` in `.env` and restart the API.

Q: Why is my output marked invalid?

A: Validation requires `citations` (non-empty list) and `retrieved_at` (ISO timestamp).

Q: Are providers required to be free?

A: No. Cost is a field in the manifest. The router will prefer cheaper providers if scores are similar.

Q: Can I add multiple providers for the same capability?

A: Yes. The router will rank them against each other at runtime.

Q: Where do the trust scores come from?

A: From successful runs, citation pass rate, and latency percentiles.

Q: Why is routing so strict?

A: It is intentional to prevent wrong tool selection and hallucinations.

Q: How can I make routing less strict?

A: Relax `MIN_RELEVANCE_SCORE` or `MIN_TOTAL_SCORE` cautiously.

Q: How do I run tests?

A: `pytest -q` in the repo root.

Common Mistakes

- Missing citations in provider responses → execution fails.
- Wrong capability tags → router rejects the app.
- No Ollama running → `--auto-caps` fails or times out.
- Over-strict thresholds → frequent `NO_MATCH`. Relax `MIN_RELEVANCE_SCORE` if needed.

Contributor Notes

- Add providers by creating a manifest + endpoint.
- Keep outputs structured JSON with citations.
- Use the tests in `tests/` for regression coverage.

License

See `LICENSE`.