**Semantis AI — Product Requirements Document (PRD)**

**Product:** Semantic Caching Platform (SaaS)  
**Repo:** Semantic\_ai (monorepo)  
**Folders:** Frontend/ (Bolt AI), Backend/ (Cursor AI)  
**Primary users:** AI developers and platform teams integrating LLMs (OpenAI, Anthropic, Bedrock, Mistral)

**0) Executive Summary**

Semantis AI reduces LLM cost and latency by caching **semantically equivalent** prompts and reusing answers. It provides:

* Drop-in API compatible with OpenAI-style endpoints.
* Hybrid cache (exact+semantic), adaptive similarity per tenant, TTL & invalidation.
* Developer SDKs (Python/Node), real-time metrics, logs, and dashboards.
* Multi-tenant isolation, security, and observability.

**North-star KPI:** Token cost saved & p50/p95 latency reduction.

**1) Goals & Non-Goals**

**1.1 Goals**

* Deliver a **developer-first, zero-infra** caching layer.
* Achieve **50–70% cache hit** on typical support/RAG workloads within 30 days of usage.
* Provide **transparent observability** (hit ratio, similarity score, savings).
* Support **multi-tenant** isolation and API-key auth.
* Offer **pluggable vector backends** (FAISS → pgvector/Redis/Milvus).

**1.2 Non-Goals (v1)**

* No end-user auth flows (only tenant API keys).
* No fine-tuning or re-ranking model training (roadmap).
* No billing/stripe integration (v2).

**2) Personas & Use Cases**

**Persona A – SaaS Dev:** Wants a one-liner wrapper around OpenAI to cut costs.  
**Persona B – Platform/ML Engineer:** Needs observability, knobs (thresholds, TTL), and audit logs.  
**Persona C – Data Privacy Lead:** Needs isolation guarantees and retention controls.

**Key use cases**

1. Wrap existing LLM calls with our SDK and see instant hits on similar prompts.
2. View dashboard: hit%, $ saved, latency, top queries, “why cached”.
3. Adjust similarity threshold and TTL per tenant.
4. Export logs & metrics to SIEM/Prometheus.

**3) System Overview**

**High-level flow:**  
Frontend (dashboard & docs) ⇄ Backend API (FastAPI) ⇄ Cache Engine (exact hash + vector index) ⇄ LLM Provider  
Storage: Vector index (FAISS/pgvector), Postgres (events/metrics), files (logs).

**Tenancy:** All data, vectors, metrics are namespaced by tenant\_id.

**4) Backend PRD (Cursor AI)**

**4.1 Functional Requirements**

* **FR-B1: OpenAI-compatible Query**
  + POST /v1/chat/completions (and simplified GET /query for dev)
  + Request body: model, messages, optional temperature, metadata.
  + Response body mirrors OpenAI style with additional meta block:
    - meta.hit: exact|semantic|miss
    - meta.similarity: 0.0–1.0
    - meta.latency\_ms
    - meta.cache\_key, meta.strategy
* **FR-B2: Exact Cache**
  + Redis hash or in-memory dict (v1) keyed by normalized prompt and model.
* **FR-B3: Semantic Cache**
  + Embeddings model: text-embedding-3-large (fallback bge-large, E5).
  + Cosine similarity via normalized vectors.
  + ANN index: FAISS (v1) → pgvector/Redis Vector (v1.1+).
* **FR-B4: Hybrid Decision**
  + Check exact → if miss: vector search top-k=5 → choose 1 best.
  + Decision rule: max\_sim ≥ θ\_client → semantic hit; else miss.
* **FR-B5: TTL & Invalidation**
  + Default TTL: 7 days.
  + Popularity extension: if use\_count ≥ N → extend TTL to 30 days.
  + Admin endpoints to invalidate by substring, tag, or key.
* **FR-B6: Adaptive Threshold**
  + Per-tenant θ\_client starts at 0.83.
  + Every 100 requests adjust ±0.01 toward target hit ratio window [0.55, 0.85].
* **FR-B7: Logging & Metrics**
  + Logs files: access.log, errors.log, semantic\_ops.log (rotating).
  + Metrics: requests, hits, semantic\_hits, misses, hit\_ratio, p50/p95 latency, tokens saved (est.), similarity distribution histogram.
* **FR-B8: Multi-Tenant Security**
  + API keys: sc-{tenant}-{uuid}.
  + Middleware extracts tenant from key; reject invalid.
  + Namespace isolation: per-tenant indexes and stores.
* **FR-B9: Observability Endpoints**
  + GET /metrics → per-tenant counters, ratios, last N ops sample.
  + GET /events?limit=100 → recent cache decisions.
* **FR-B10: Health & Readiness**
  + GET /health returns "ok" and backend versions.
* **FR-B11: SDK Support**
  + **Python/Node** examples; tolerates network errors and retries (exponential backoff).

**4.2 API Design**

**4.2.1 Public Endpoints (v1)**

* POST /v1/chat/completions
  + **Headers:** Authorization: Bearer sc-...
  + **Body (subset OpenAI):**
  + {
  + "model": "gpt-4o-mini",
  + "messages": [{"role":"user","content":"Explain inflation"}],
  + "temperature": 0.2,
  + "metadata": {"tags":["support"],"domain":"finance"}
  + }
  + **200 Response:**
  + {
  + "id":"chatcmpl-...","object":"chat.completion","created":...,
  + "model":"gpt-4o-mini",
  + "choices":[{"index":0,"message":{"role":"assistant","content":"..."},"finish\_reason":"stop"}],
  + "usage":{"prompt\_tokens":..., "completion\_tokens":..., "total\_tokens":...},
  + "meta":{"hit":"semantic","similarity":0.91,"latency\_ms":84,"strategy":"hybrid"}
  + }
* GET /metrics
  + Returns counters and rolling windows.
* POST /admin/invalidate (internal / future enterprise)
  + Body: {"match":"substring"|"regex":"...","tenant":"devA"}
* POST /admin/threshold (internal / future enterprise)
  + Body: {"tenant":"devA","theta":0.86}

**4.2.2 Internal Endpoints (operator-only)**

* GET /health
* GET /events?tenant=...&limit=...

**4.3 Data Models**

**CacheEntry**

* id, tenant\_id, prompt\_norm, response\_text, model, embedding (vector),  
  similarity\_score, strategy (exact|semantic|miss), ttl\_seconds,  
  created\_at, last\_used\_at, use\_count, domain\_hint, metadata JSONB.

**Event**

* timestamp, tenant\_id, prompt\_hash, decision, similarity, latency\_ms.

**Metric rollups**

* Per minute/hour/day aggregates: requests, hit%, semantic%, p50/p95 latency, cost saved.

**4.4 Logging**

* **access.log**: ts tenant method path status latency\_ms
* **semantic\_ops.log**: ts tenant decision sim model key use\_count ttl
* **errors.log**: stack traces, request context, tenant.

Rotation: 5MB, 3 backups. Optional ship to CloudWatch/Stackdriver.

**4.5 Non-Functional Requirements (Backend)**

* **Performance**:
  + p50 latency: < 150ms on cache hit; p95 < 350ms (excluding provider latency).
  + Throughput: 200 RPS on medium node (scales horizontally).
* **Availability**: 99.5% (v1), health checks & readiness.
* **Security**: HTTPS only, API keys, input size limits, rate limiting per tenant.
* **Scalability**: Stateless API layer; pluggable vector store.
* **Privacy**: Tenant isolation; configurable retention, optional encryption at rest (backlog).
* **Compatibility**: OpenAI-style request/response; Python 3.10+, Node 18+.

**4.6 Acceptance Criteria (Backend)**

* 100% of requests return structured meta with hit/exact/semantic/miss.
* Exact cache hit for identical prompt within session.
* Semantic hit fires for paraphrases with sim ≥ θ\_client.
* Logs present for every request and error.
* Metrics endpoint returns accurate counters vs. logs sample.
* Admin invalidation removes entries and rebuilds index without crash.

**5) Frontend PRD (Bolt AI)**

**5.1 Functional Requirements**

* **FR-F1: Authentication UI (Tenant key input)**
  + Developer pastes API key; persisted in local storage (never logged).
* **FR-F2: Query Playground**
  + Text area to enter prompt and choose model/temperature.
  + Shows response and **meta panel** (hit type, similarity, latency).
  + Allows re-run and A/B compare (cache vs fresh).
* **FR-F3: Metrics Dashboard**
  + Cards: Hit ratio, semantic hit ratio, avg latency, estimated $ saved.
  + Charts: time-series of hits/misses; histogram of similarity scores.
  + Top queries & clusters (tooltips: “why cached”).
* **FR-F4: Settings**
  + Set per-tenant: θ (similarity threshold), default TTL, max hit age.
  + Toggle domains/tags (informative only in v1).
* **FR-F5: Logs Viewer**
  + Streams last N semantic events (decision, sim, latency).
  + Download CSV button.
* **FR-F6: Docs & SDK**
  + Copy-paste snippets for Python/Node.
  + Example curl calls with current tenant key.
* **FR-F7: Health Indicator**
  + Shows backend status (/health) and endpoint URL.

**5.2 Information Architecture (Pages)**

* / → Overview/KPIs
* /playground → Query tester
* /metrics → Time-series & aggregates
* /logs → Events table (virtualized list)
* /settings → Tenant config (θ, TTL)
* /docs → SDK + examples

**5.3 UI/UX Requirements**

* Clean, developer-first (dark theme default).
* Latency & hit type badges on results.
* Copy-to-clipboard buttons for API calls & keys.
* Empty state hints for new tenants (seed examples).

**5.4 Frontend API Contracts**

* **Env vars**
  + VITE\_BACKEND\_URL, SEMANTIC\_API\_KEY
* **Calls**
  + GET {BACKEND}/metrics (Authorization: Bearer key)
  + POST {BACKEND}/v1/chat/completions with body (OpenAI-style)
  + GET {BACKEND}/events?limit=100

**5.5 Components (example)**

* components/KpiCards.tsx: hit%, latency, savings.
* components/SimilarityChart.tsx: histogram from /events.
* components/Playground.tsx: input, run, results meta.
* components/LogsTable.tsx: virtualized, filter by decision.
* hooks/useSemanticCache.ts: wraps API calls & error handling.
* api/semanticAPI.ts: fetch helpers with auth header injection.

**5.6 State & Telemetry**

* **State:** prompt, result, last meta, timeline data (10–50 points).
* **Events logged to backend:** ui.query.run, ui.settings.save, ui.docs.copy\_snippet.

**5.7 Non-Functional (Frontend)**

* Initial load < 2s on broadband.
* Charts render < 300ms for last 1k events.
* No PII stored; only tenant key in local storage by consent.
* Errors surfaced with actionable message; no stack traces to user.

**5.8 Acceptance Criteria (Frontend)**

* Query playground displays hit/miss & similarity for every call.
* Metrics & logs update after each call (pull-based).
* Settings change persists and affects next queries (if backend supports).
* Docs page shows working curl, Python, Node snippets with **masked key**.

**6) Data Retention & Compliance**

* Default retention: 30 days of logs & events (configurable).
* Cache entry TTL default 7 days (extend on popularity).
* “Purge tenant” operator action removes vectors, events, logs for tenant.
* Optional “no-store” flag on request (don’t cache/store this call).

**7) Security**

* HTTPS only; HSTS if managed edge.
* API-key auth (Authorization: Bearer sc-...).
* Rate limit per tenant (e.g., 60 RPS burst, 600 RPM sustained).
* Input size limit: 20k chars/request (config).
* CORS: allow-list tenant domains (optional).
* Secrets management: .env for local, platform secrets for cloud.

**8) Deployment & Environments**

* **Local dev:** uvicorn, Vite; .env files in each folder.
* **Staging:** Railway/Render (single container), Supabase (pg + pgvector) once enabled.
* **Prod:** API behind managed load balancer; autoscaling pods; managed Postgres/Redis; log shipping to CloudWatch/Grafana.

**Monorepo CI/CD (GitHub Actions)**

* Lint & test.
* Build docker images for Backend; build & deploy static frontend.
* Tag releases: backend-v0.1.0, frontend-v0.1.0.

**9) KPIs & SLAs**

* **KPIs**
  + Hit ratio (overall & semantic), $ saved/day, p50/p95 latency.
  + Time-to-first-value (TTFV): time from first call to first semantic hit.
  + Error rate < 0.3%.
* **SLA (v1 target)**
  + 99.5% monthly API availability.
  + Support response < 24h on business days.

**10) Test Plan (sample)**

**Backend**

* Exact hit: identical prompt → hit=exact.
* Semantic hit: paraphrase → hit=semantic, sim ≥ θ.
* Miss path: new prompt → LLM called; entry stored.
* TTL expiry: advance clock → entry pruned; index rebuilt safely.
* Invalidation: remove by substring; confirm no retrieval.
* Adaptive threshold: synthetic traffic reduces/increases θ.

**Frontend**

* Playground: run prompt → result & meta visible.
* Metrics reflect last calls; logs table grows.
* Settings update persists and reflected in server (stub if v1).
* Docs snippets execute successfully (curl & SDK).

**11) Roadmap**

**v1.1**

* pgvector (Supabase) or Redis Vector backend.
* POST /feedback {good|bad} → supervised threshold nudging.
* Export metrics to Prometheus.

**v1.2**

* Billing & usage (Stripe) with per-token savings estimation.
* Public API docs (Docusaurus) and examples gallery.
* LangChain/LlamaIndex “cache” plugin.

**v2**

* Domain-aware multi-embedding router (FinBERT, PubMedBERT).
* Enterprise SSO (SAML/OIDC), SOC2 readiness.
* Data masking & PII scrubbing.

**12) Repository Blueprint**

Semantic\_ai/

├── Backend/

│ ├── semantic\_cache\_server.py

│ ├── logs/ (gitignored)

│ ├── requirements.txt

│ ├── Dockerfile

│ └── README.md

├── Frontend/

│ ├── src/

│ │ ├── components/

│ │ ├── hooks/useSemanticCache.ts

│ │ └── api/semanticAPI.ts

│ ├── public/

│ ├── package.json

│ ├── vite.config.ts

│ ├── .env.example

│ └── README.md

└── PRD/

└── Semantis\_AI\_PRD.md

**13) Acceptance & Launch Checklist**

* Backend endpoints live with API-key auth; healthcheck OK.
* Logs rotating and readable; metrics consistent with logs.
* Frontend playground returns answers with meta in < 2s on local.
* Readme includes **Quick Start** (copy-paste).
* Sample SDKs validated.
* Demo script: run 10 paraphrases → show 60%+ semantic hits and $ savings estimate.

**Appendix A — Sample Error Codes**

* 401\_UNAUTHORIZED: missing/invalid API key.
* 429\_RATE\_LIMITED: per-tenant throttle.
* 413\_PAYLOAD\_TOO\_LARGE: prompt exceeds allowed size.
* 500\_PROVIDER\_ERROR: upstream LLM error (return diagnostic id).

**Appendix B — Savings Estimation**

* Track usage.prompt\_tokens & usage.completion\_tokens on **miss** path.
* Multiply by provider price → baseline\_cost.
* On **hit**, savings = expected cost of equivalent miss.
* Dashboard aggregates per day/week/month.