ALEKSANDR TIKHONOV.

AI SYSTEMS ENGINEER · LLM AGENT ARCHITECT

I design and operate production systems around LLMs: RAG services, agent orchestration, semantic search, and self-hosted inference — under strict latency and cost constraints.

CONTACTS	
Location	Astana, Kazakhstan
Email	ascorblack@gmail.com
Email	a@scorblack.ru
Telegram	@notsoulmate
GitHub	github.com/ascorblack
Labs	github.com/ascorblack-labs
Protocore	protocore.ascorblack.com

2.5+
YEARS OF PRODUCTION AI

20
UNIVERSITIES PILOTING LANGPT

11,000+
TESTS IN PROTOCORE

MPL-2.0
OPEN PROTOCORE CORE

01 PROFILE

AI Systems Engineer / LLM Agent Architect with two and a half years of commercial production experience in LLM/RAG systems, AI backend infrastructure, semantic search, recommender systems, local LLM inference, and agent orchestration.

Since January 9, 2024 I have worked at EBS Lan LLC (a Russian academic e-library platform) under an official employment contract (contract title — ML engineer; actual role — AI Systems Engineer / LLM Agent Architect). I own the backend and infrastructure implementation of the company's key AI services: architecture, development, integration, deployment, monitoring, and production support of educational AI products.

In parallel — founder of ascorblack-labs: Protocore, a protocol-first runtime for LLM agents. Its core is open under MPL-2.0 and installs via pip; the platform layer (enterprise backend, sandbox, chat, dashboard) is proprietary and runs entirely on my own infrastructure.

My strength is engineering production systems around LLMs rather than generic model training: service design, RAG/backend architecture, integrations, resilience, observability, and operating AI in real infrastructure under strict latency and cost constraints.

02 EXPERIENCE

EBS Lan LLC

ML engineer (per contract) · AI Systems Engineer / LLM Agent Architect (actual role)

`01.2024 – present`

I own the backend and infrastructure implementation of the company's key AI services: architecture, development, integration, deployment, monitoring, and production support.

01 LanGPT — production RAG/LLM assistant for academic tasks

- Built the backend side of the product: the public API, the RAG service, the LLM backend balancer, and the administration components; I take part in their deployment and maintenance in Kubernetes.
- Public API with JWT authentication, rate limiting by pricing plan, and SSE streaming of responses; generation tasks run through RabbitMQ.
- RAG service over Elasticsearch for verified educational and academic content from the e-library, available to the company through partner agreements with publishers, without reaching out to the external internet.
- LLM backend balancer with a dynamic server registry, least-loaded / least-connections selection with priority, fallback, and OpenAI-compatible adapters.
- The AI processing pipeline covers more than ten academic-writing tasks: hypothesis formulation, summarization, drafting sections with citations, content checking; includes content moderation of user requests.
- Production monitoring on Prometheus, Grafana, and Elasticsearch.

`20 universities in pilot` `37,000+ requests` `6,500 active users`

Figures come from the company's public materials. The role and area of responsibility can be confirmed by the employer on official request.

02 AI-Snippets — generating search snippets over EBS Lan search results

- Pipeline: two-level cache (exact + semantic), query classification and expansion, multi-source search over Elasticsearch, context assembly (BM25 + RRF), generation on vLLM with a final quality gate.
- Polling scheme via Redis, circuit breaker, and graceful degradation: on any problem the snippet is simply not shown, with no error surfaced.
- Tuning of vLLM parameters for snippet performance on a specific server configuration.

`stress test 13.2 h · 55,200 requests · 0 × 5xx` `quality-gate OK rate 87.3%`

`median 4.6 s against a 5 s SLA`

03 Search 2.0 — EBS Lan search platform

- A platform of several production services: the main search API, the subject-area and author classifier, a vectorizer with caching, a spell checker, and search over educational video.
- Hybrid BM25 and vector search over Elasticsearch with a custom coefficient system and boosts by relevance and year of publication.
- All ML components are designed CPU-only — no GPU dependency and strict resource limits.

`hybrid BM25 + kNN + RRF` `ML: CPU-only`

04 Books Keywords Generator — automatic catalog enrichment

- Two-stage LLM generation of keywords for e-library books from the table of contents — generation, then validation by a separate prompt checking language, length, uniqueness, and meaningfulness.
- Dynamic context-length management, parallel processing with GPU overload protection; structured JSON logs in Elasticsearch.

03 INDEPENDENT PROJECTS

01 Protocore — protocol-first runtime for LLM agents · founder / solo developer

protocore.ascorblack.com ↗

The core is "the agent loop, and nothing else": the ReAct runtime, the context budget, the tool surface, compaction, and stop conditions. Everything outward-facing is a Protocol the host implements. Open under MPL-2.0 (ascorblack-labs/protocore-community, `pip install protocore`). The platform on top of the core is proprietary — enterprise backend, sandbox, chat, dashboard, autonomous worker, Helm/Kubernetes deployment — designed and built by me end to end.

- 20 Protocol interfaces (ILLMProvider, IRunStore, IToolRegistry, IMemory, IWorkspace…), zero upward imports — the core knows nothing about databases, HTTP, or deployment; direct/deep run strategies, delegation to subagents with width and depth budgets, snapshot and resume of runs.
- Three-layer tool surface (tenant policy → clipped surface → progressive discovery over BM25) and multi-tier context compaction; 500+ runtime constants in a frozen per-tenant snapshot.
- Enterprise layer: REST API with SSE streaming, RBAC, access plans with quotas and audit, sandboxed tool execution (gVisor), observability, and an autonomous worker for background tasks.
- Stack — FastAPI, Next.js 15, PostgreSQL, Valkey (Redis), RabbitMQ, OpenSearch, Helm / Kubernetes; all infrastructure is self-owned: GitLab, Harbor, k3s.
- A reproducible-research and publication site (protocore.ascorblack.com): methodology, limitations, data, and PDF papers.

`11,000+ tests across core and enterprise`

`core: 90% branch coverage, strict typing, Python 3.12–3.14`

`ruff / mypy / bandit / pip-audit — clean`

02 Daedalus — personal self-developing agent in Telegram · MIT

github.com/ascorblack/daedalus ↗

An agent on the Protocore core that lives in Telegram and in its own web app, runs inside a Linux container with real tools, and changes its own code through pull requests approved right from the chat.

- One operator, one container: shell, browser, git, filesystem, long-running services, a scheduler, subagents, memory; each Telegram forum topic is a session with its own workspace.
- Built to run for weeks: sessions survive restarts, runs resume from snapshots, context is compacted instead of lost, spend is capped by a supervisor the agent cannot edit.
- React 19 web app (PWA): live transcripts, a file browser with previews, two sessions side by side; self-hosted SearXNG for web search, MCP servers per session.

`320+ tests` `Python 3.12 · Docker Compose · React 19`

03 swiftclf-tuna — AI-orchestrated research · bilingual intent classification

github.com/ascorblack/swiftclf-tuna-research ↗

Hierarchical (RU/EN) intent classification for routing AI assistants. My role — problem framing, taxonomy and evaluation-criteria design, directing the agentic workflow, and human-in-the-loop validation of results.

- Taxonomy of 4 L1 / 22 L2 classes; a reproducible holdout set of 2,625 examples; the metrics are a verifiable applied result, not a SOTA claim.
- Selective prediction with a fallback branch and risk-aware metrics; probability calibration; CPU-first deployment with model cards, training / eval pipelines, and holdout reports.

`accepted accuracy 87.07%` `ECE 0.061 · coverage @ risk ≤ 0.14 — 87.24%`

`278M parameters · p95 77 ms / 1 CPU core`

04 Open source: tools for AI agents — grown out of daily work with autonomous agents

github.com/ascorblack ↗

- ai-audit-kit — a portable `.audit` standard: a reproducible AI code-audit pipeline (audit → triage → fix → review).
- notify-telegram-cli — Telegram notifications from autonomous agents; pure standard library, zero dependencies.
- kb-genesis — bootstrapping `.kb/` knowledge bases for agent environments (Claude Code, Codex, Gemini CLI, Cursor).
- LightUniLLM — a lightweight LLM framework over the OpenAI Responses API: Pydantic, structured outputs, streaming.

---

04 TECHNOLOGY STACK

01 LLM & AI SYSTEMS	02 BACKEND
• LLM services, RAG, agent orchestration • Local inference, vLLM tuning (max-num-seqs, KV cache, prefix-aware prompt design, speculative decoding) • OpenAI-compatible adapters, dynamic provider registry • Prompt & runtime design, tool routing, context compaction • Selective prediction, calibration, quality gates, logprobs-based confidence	• Python, FastAPI, asyncio, Pydantic v2 • SSE streaming, JWT, REST APIs • Prioritized task queues, SSE session recovery (Redis Pub/Sub + TTL cache), graceful degradation • Microservices, background workers, pipeline architecture

03 DATA & SEARCH

- Elasticsearch / OpenSearch: BM25, kNN, hybrids with RRF
- Semantic reranking, sentence-transformers, multilingual encoders
- PostgreSQL + pgvector, MariaDB / MySQL, Redis / Valkey, RabbitMQ
- Recommender systems, semantic profiling

04 INFRASTRUCTURE & OPS

- Docker, Nginx, Kubernetes / Helm, k3s, gVisor
- Self-hosted: GitLab CI, Harbor, cert-manager
- Prometheus, Grafana, structured logging, OTLP
- Latency- and cost-aware design, observability and debuggability in production

05 FRONTEND

- Next.js 15, React 19, TypeScript — platform frontends (chat, dashboard, PWA)

06 AI-ORCHESTRATED RESEARCH & ENGINEERING

- Problem framing, methodology design, and result validation for autonomous AI-agent workflows
- Human-in-the-loop review, counter-review, reproducible reports
- Model cards, holdout evaluations, risk-aware reports, negative results as part of the process

05 LANGUAGES

Russian	native
English	technical reading / listening; written and spoken communication — developing