OMAR MOSTAFA

omar.mostafa0205@email.com | github.com/omar-mostafa205 | Portfolio | LinkedIn

PROFESSIONAL SUMMARY

AI engineer and founding (first) engineering hire at a healthcare AI startup, sole builder of the core agent and matching layer for a live B2B platform serving 1,000+ practices. Shipped a hybrid retrieval and ranking engine — OpenAI embeddings indexed in pgvector, fused with weighted full-text search and an 11-tier multi-signal scorer, sharpened by a structured feedback-capture loop on every LLM output. On top of that: a multi-tool agent system far beyond a single prompt in a loop — an orchestrator routing to specialist tools, LLM proposals bounded server-side and solved via constrained optimization (two-pass linear programming), governed by a three-layer eval suite (deterministic gates, agent-driving-agent simulation, LLM-as-judge) that decides what the agent may do alone. Sole engineer owning reliability end-to-end for a two-sided platform. Strong Python and TypeScript; multiple products taken from zero to production solo and kept running.

EXPERIENCE

Founding Engineer (First Engineering Hire) — tp53 health

Remote *Jan 2026*

- Built production **agentic AI infrastructure**: a multi-tool clinical planning agent on **Anthropic (Claude) tool use** — model proposals are macro-classified and bounded server-side, then solved by a **two-pass linear program** (feasibility bands + slack-penalized meal distribution; L1 closest-point fallback returning repair hints the model uses to self-correct) on every generated plan
- Designed the **three-layer evaluation suite that gates agent autonomy**: deterministic safety gates, agent-driving-agent simulation, and LLM-as-judge scoring on a production-calibrated rubric — findings shipped as enforcement code (deterministic allergen/diet backstop, solver-verified target verdicts); surfaced an agent that could persist fabricated clinical data after unreadable attachments, before it reached users
- Shipped **retrieval, ranking, and matching in production**: LLM-powered document ingestion (extraction, classification, tagging), recursive token-budget chunking, weighted Postgres full-text indexes (tsvector/GIN), and tiered top-k retrieval with rank-merge and dedupe exposed as LLM tools; an 11-tier relevance scorer serving both the product UI and agent tool calls; Levenshtein-based entity resolution pairing free-text model references to records
- Shipped **structured per-response feedback capture** on LLM outputs (rating + categorized failure reason + free text, persisted per-message) — the raw signal for ranking/eval loops; LLM extraction mapping free-form consultation transcripts onto structured intake-form fields with clinician review; budget-bounded, schema-validated context assembly for PHI-bearing chat
- **Sole engineer owning production reliability and velocity** for 1,000+ active patient practices: patient record workflows, Stripe billing, transactional email (Resend), PDF generation; diagnosed a production latency regression traced to excessive sequential model round-trips
- Established engineering standards from scratch: code review discipline, an enforced engineering rulebook, CI, and observability
- Stack: **Python** (ML & data pipelines), **Next.js 16, TypeScript, Node.js, Prisma, Postgres (Neon)**, Vercel

PROJECT EXPERIENCE

AscendAI - AI Interviewing Platform

Source code | *App*

Next.js, Node.js, WebSockets, TypeScript, PostgreSQL, Redis, BullMQ, Sentry, Docker, GitHub Actions

- Built and run a full-stack platform where **AI agents hold live voice conversations with real users** — streaming STT + TTS via Gemini Live over WebSockets at sub-200ms latency
- Implemented LLM feedback pipelines with Redis + BullMQ background workers and batched persistence, cutting processing time by 45%
- Production-grade operations: Sentry monitoring, Redis sliding-window rate limiting, JWT/RBAC auth — **<0.1% unhandled error rate**; Dockerized with zero-downtime CI/CD

Healthcare Fraud Detection (Machine Learning) *Source code* | *App*

Python, scikit-learn, pandas, NumPy

- Built an end-to-end **Python scoring pipeline**: preprocessing, feature engineering, training, and evaluation across multi-source datasets (inpatient, outpatient, beneficiary)
- Engineered 55+ features; feature selection via correlation analysis and model-based importance cut the feature space >50% for inference efficiency
- Trained and tuned classification/scoring models (Logistic Regression, Decision Tree, Gradient Boosting): **95% accuracy**

Cybership — SaaS Landing Page & Design System *Source code* | *App*

Next.js, TypeScript, TailwindCSS, Framer Motion

- Engineered a **production-grade SaaS landing page** with **advanced animation systems and micro-interactions**, achieving smooth **60fps motion** across all device sizes
- Architected a **scalable component system** with **conversion-focused UI patterns** — modular, reusable, and designed for rapid iteration
  Implemented **performance-optimized rendering strategies** (SSG, lazy loading, **Core Web Vitals** tuning) for fast load times and strong SEO fundamentals
- Demonstrated **end-to-end product thinking**: from visual design system to deployment-ready frontend infrastructure

Recurrly - Subscription Tracking Mobile App *Source code* | *App*

Expo, React Native, TypeScript, Node.js, Express, MongoDB, Clerk, Upstash QStash, PostHog, NativeWind

- Owned the product **end-to-end** as a solo engineer — from **API design and auth** to **dashboard UI, analytics integration**, and **automated workflows** — shipping a fully functional app from zero to production
- Built a **React Native + TypeScript** frontend with a subscription dashboard featuring renewal tracking, insights, and real-time status updates, wiring **PostHog analytics** for user behavior and product usage monitoring
- Architected a **Node.js + Express REST API** with **Clerk JWT auth**, **MongoDB** persistence, and a fully documented **OpenAPI spec** — designed for clean integration and rapid iteration in a startup environment
- Designed an **event-driven notification workflow** using **Upstash QStash** that schedules and delivers multi-step reminder emails (7/5/2/1/0 days before renewal) via **Nodemailer**, demonstrating ownership of async backend flows end-to-end

Docy - AI-Powered Codebases Documentation Generator *Source code* | *App*

Next.js, TypeScript, PostgreSQL, Prisma ORM, NextAuth.js, tRPC, Google Gemini AI, Zod

- Developed a full-stack documentation generator powered by **Google Gemini AI** with a custom **AST code analysis engine**, supporting OAuth via **NextAuth.js** (GitHub, Google, GitLab)
- Architected **PostgreSQL schema with Prisma ORM** and **tRPC backend**, implementing B-tree and **full-text search indexes (tsvector)** — reducing query time by **70%**
- Implemented **Next.js server-side caching** with request memoization, cutting **database load by 60%** and API response times from **400ms → 80ms**
- Designed **tRPC backend** with **Zod input validation**, typed API contracts, and protected mutation endpoints enforcing authentication middleware

TECHNICAL SKILLS

AI/ML: Agent systems & orchestration (multi-tool loops, tool use), agent evaluation (deterministic gates, trajectory assertions, LLM-as-judge), RAG & embeddings retrieval, document ingestion, LLM APIs (Anthropic/Claude, OpenAI, Gemini), prompt engineering, scikit-learn

Backend: Python (ML & data pipelines), Node.js, REST APIs, tRPC, async patterns; FastAPI & Pydantic AI (actively deepening), schema validation (Zod / Pydantic-style)

Frontend: React, TypeScript, Next.js, Tailwind CSS, shadcn/ui

Data: PostgreSQL (indexing, full-text search), SQL, Redis, MongoDB Atlas

Infrastructure: AWS, Vercel, Docker, CI/CD (GitHub Actions), Sentry observability; OAuth 2.0, JWT, RBAC

EDUCATION

German International University (GIU) Graduation Date: 2026

Bachelor of Computer Science