The most extraordinary thing I've worked on.

The term extraordinary is quite relative and wide. Here's how I would define extraordinary: solving a problem that is highly relevant in today's day and time with the resources at hand.

In the last couple years when AI blew up, I noticed that as a developer there are no pain-free ways of working with LLMs at scale - since I am an engineer and wanted to integrate LLMs into my apps. Emphasis on "pain-free" - because sure there are tools such as Ollama-cpp, AWS, GCP and so on. However, the time, resources, and capital that developers invested in building infrastructure to create well-optimized, cost-efficient LLMs was the same, if not more than the savings gained from those models.

That's why I built model0 - a one-stop solution for all LLM, and ML devs. It's a platform as a service that utilizes AWS SageMaker and Bedrock to setup all the necessary infrastructure for any model that the developer needs, carry all the required optimizations on it (pruning, quantizations, RAG) and get a singular deployment link that the developer can integrate into their app.

Think Vercel, but for LLM infrastructure. The core philosophy is to reduce clutter, time and capital throughout the entire process so businesses can focus on the creative aspect of things. We use AWS SageMaker's infrastructure to train, prune and quantize models pulled from Huggingface, and host it on Bedrock - since it's much more cost effective for LLM inference as compared to real-time SageMaker.

How hard did you work on it? Would you do it again?

I have been working on it since October last year. I was in my second year of grad school, so managing school, being a TA and shipping on model-0 was tricky - but I'd do it all over again. Although since I've started, I've re-iterated on my core approach multiple times. I initially began with having only a SageMaker infrastructure, but then after initial customer reviews - I gathered critiques that the pricing goes too high. I love shipping things quickly - the faster I ship, the faster I get customer feedback, the faster I know if a feature worked or not, the faster I iterate.

I believe iteration is the most important thing that determines the success of a product. I like to call this the "momentum" of a team. One should get attached to the product, not the feature because it's the features that make a product, not vice-versa.

I'd love to talk more about what Entangl's working on and how I can contribute. In the meantime, know more about me on the links below:

Portfolio @ https://athk.dev
X @ https://x.com/athkdev
GitHub @ https://github.com/athkdev
LinkedIn @ https://linkedin.com/in/athk

Atharva Deepak Kamble

me@athk.dev

Full Stack Engineer • GitHub • LinkedIn • Portfolio • Open for relocation across the US 857-200-4661

Education

Northeastern University — Boston, MA
Master of Science, CS + Information Systems (full stack infrastructure emphasis) — *Sep 2023 - May 2025*
Relevant courses: Algorithms, Linux/UNIX, Web and API development, Object Oriented Design in Java, Design patterns, Operating Systems

University of Mumbai — Mumbai, IN
Bachelor of Engineering, Electrical Engineering & Computer Science — *Aug 2018 - May 2022*
Relevant courses: Programming in C, Object Oriented Design, Databases, Computer Networking & Architecture, Big Data, Map Reduce, Hadoop, Image Processing

Technical Skills

Frontend	React, Typescript, JavaScript, Next.js, Vue.js, Vite, npm, JQuery, Tailwind, Three.js
DevOps & Systems	Linux (Fedora and Ubuntu), Bash, Docker, Jenkins, Terraform, CI/CD, GitHub Actions, REST APIs, Cloudflare, Git
Backend & Databases	Python, Django, Go, GraphQL, C++, SQL, PostgreSQL, MVC API design, Bash, npm, HTTP, TCP/IP
Data & AI	ETL Pipelines, AWS EC2, AWS SageMaker, LangChain, OpenAI, Pandas, Numpy, Matplotlib, Plotly, PyTorch, Apache Kafka

Work Experience

Northeastern University — Boston, MA
Graduate TA — *Sep 2024 - Apr 2025*

- Instruct students in Angular, Typescript, Git, GitHub and the Agile methodology through 1-on-1 sessions and online communications.
- Run weekly office hours, breaking down tough concepts and debugging live for 25+ students, including pair programming when necessary.

Cognizant — Mumbai, IN
Software Engineer 1 — *Aug 2022 - Jun 2023*

- Built and maintained an in-house Angular component library with **28+** reusable and themeable UI elements. Used by **5+ teams** to improve development speed and maintain visual consistency.
- Developed pixel perfect UI as per wireframes from the UX design team for OneCognizant (company-wide appstore). Built in **React, Tailwind, SCSS and Typescript**.
- Engineered a payments app for external stakeholders by integrating Stripe using **React, Python, Apache and AWS**, supporting **40+** local payment gateways and **130+** currencies.
- Write integration tests with Playwright and React Testing Library to validate end-to-end user workflows - ensuring **90%+** test coverage.
- Refactored a monolithic app into modular **Python microservices**, enabling service-specific deployments and minimizing downtime during updates.
- Developed a well documented **Python** module that simplified application deployment workflow that automates containerization with **Docker and Terraform** resulting in a **one-click deployment solution**.

Raftlabs — Remote
Software Engineer (Founding Engineer) — *Apr 2021 - Aug 2022*

- Developed a website for a supply chain analytics tool, optimizing delivery schedules and inventory management with scalable **React, and Java.** Used the **in-browser IndexedDB** for local caching resulting in extremely fast user experience.
- Revamped a financial loyalty app for the Bank of Ireland using **React, GraphQL, Python (Django), and Docker** to streamline revenue tracking and help customers gain rewards.
- Integrated Meta's webhooks with **Shopify and Node.js** to help viewers buy products by commenting codes in **Facebook live streams**.
- Coded a **proof of concept** for an efficient coupon distribution system in Typescript, Node.js and PostgreSQL that could handle **2000 requests per second**.
- Implemented an automated CI/CD pipeline using **GitHub Actions and Python** to schedule blog posts for the marketing team - resulting in an efficient workflow with no manual overhead.

Projects (Infrastructure and full stack)

Model-0 - Platform as a Service — github.com/athkdev/model0

- Full stack web app for creating and deploying AI chatbots with AWS SageMaker providing integration for developers **in less than 4 minutes.**
- Set up HTTP Apache server with Amazon EC2 instance to handle swift load balancing and secure encryption by setting up a SSL certificate.
- Developed a **robust REST API** with authenticated routes for communicating with SageMaker API to manage creation, and tuning of ML models. Create permissions in Linux for Apache server by adding it to the current user group.
- Configured a reverse proxy between the Apache route traffic from web and localhost for serving a dynamic Next.js app.
- Built a GitHub actions CI/CD pipeline that deploys code on an EC2 instance automatically once code is pushed to GitHub.
- Used **Next.js, React.js, REST API, Typescript, PostgreSQL AWS SageMaker and AWS EC2**. The app is launched live here.

Mental Health Counseling AI Chatbot — github.com/athkdev/legacytheapp

- Chatbot-powered full stack app designed for mental health counselors to make informed decisions while actively counseling.
- Extracted NLP insights from a dataset of **3000+ counsellor-client** sessions and stored them in **PostgreSQL database** for filtering and pagination.
- Integrated an OpenAI-powered chat that is pre-trained to respond like a counsellor with guided conversations to improve accessibility.
- Built with **OpenAI, Next.js, React, Typescript, Node.js, Python and PostgreSQL**. The app is launched live here.

Highly performant web crawler in Java — github.com/athkdev/crawler

- Highly performant web crawler in Java that includes a CLI and stores all URLs in a graph data structure using Neo4j.
- Applied design principles for a scalable and low-latency architecture. Store URLs in Neo4J database concurrently in a graph format.
- Utilised Java's CompletableFuture API to achieve concurrency and parallelism. Benchmarked at 730 URLs per second.
- Built with **Java and Neo4J**.