Lumina Clippers

JOB DESCRIPTION

Full-Stack Engineer

About Lumina Clippers

We help brands get more visibility through a network of creators — called clippers — who make short video clips and post them on their own TikTok, Instagram, YouTube, and X accounts. Brands fund campaigns and set a rate per thousand views. Clippers submit their post links to our platform, we track the views, likes, and comments those posts generate over time, and payouts flow from there — brands pay for the views delivered, clippers get paid for the views they drive.

The whole system runs on accurate, up-to-date metrics. Every payout decision depends on view counts being correct.

The Role

We serve different types of clients and clippers, each with their own needs and workflows. To support that, we're building a suite of internal tools tailored to how each side of the business operates. You'll be working across these tools, owning features end-to-end from database schema through API to UI.

We're a small, fast-moving team. We're looking for an engineer who can operate across the full stack without needing to be hand-held at any layer.

What You'll Build

- Tools to manage campaigns, track submissions, and surface performance metrics across internal and client-facing views
- Integrations with third-party data providers to automatically pull and refresh metrics
- Background processing pipelines for data enrichment and fraud/quality checks
- Payout and earnings tracking across multiple payment methods

---

Required Qualifications

BACKEND

- **3+ years with Python** using a modern async framework (we use FastAPI; Django REST or Flask with async are acceptable backgrounds)
- **SQL and ORM fluency** — designing relational schemas, writing efficient queries, understanding transactions and row-level locking
- **PostgreSQL in production** — connection pooling, indexing, query performance
- **Background job patterns** — polling loops, retry/backoff, graceful failure handling without a heavy queue framework
- **REST API design** — clean route structure, dependency injection, role-based access control, JWT authentication

FRONTEND

- **3+ years with TypeScript + React** — type-safe components, custom hooks, API client abstractions
- **Next.js 14+ (App Router)** — routing, server vs. client components, middleware, multi-environment config
- **Tailwind CSS** — utility-first styling, responsive layouts

INFRASTRUCTURE

- **Docker** — writing and debugging Dockerfiles, understanding multi-container setups
- **CI/CD** — reading and modifying GitHub Actions pipelines, managing staging and production promotion flows
- **Deployment platforms** — comfortable configuring environment variables, secrets, and service config across dev/staging/prod without breaking things

INTEGRATIONS

- Confident integrating and debugging third-party APIs — parsing responses, handling rate limits and failures, tracking usage and cost
- Familiar with S3-compatible cloud object storage

Strong Advantage

- Experience building data ingestion or enrichment pipelines
- LLM API integration (extracting structured data from unstructured inputs)
- Authentication systems — OAuth, JWT, session management, role-based access
- Prior work on financial data flows — earnings calculations, payout ledgers, audit trails
- Webhook and event-driven integration patterns

How We Work

- Small team, high autonomy — you'll own features from schema design to shipped UI
- We move fast and deploy often; production is always close
- Async-first communication, minimal meetings
- You're expected to make judgment calls and flag blockers early — not wait to be told what to do

---

What We're NOT Looking For

- Someone who needs a dedicated DevOps engineer for anything infra-adjacent — you should be able to debug a failed deploy or broken build yourself
- Engineers who are only comfortable in one layer of the stack — this role touches backend, frontend, workers, and third-party integrations
- Engineers who need a full test suite before feeling confident shipping — we move lean and rely on careful code review and good judgment

---

The Interview

Three stages:

1. **System design** — we'll give you a realistic data problem and ask how you'd model and build it
2. **Code review** — you'll read a real diff and tell us what you'd change and why
3. **Take-home** — a small self-contained project (3–5 hours) that mirrors the kind of work you'd do in the first month

---