Samuel Abiodun

Lagos, Nigeria | hisamuelab@gmail.com | +234 905 353 1504 | linkedin.com/in/samuelabiodun | GitHub

SUMMARY

Software/AI engineer who ships end to end — from durable, event-driven LLM pipelines and a multimodal eval framework to the AWS/Terraform infrastructure and zero-downtime deploys that run them. Helped take an AI video-generation platform from MVP to public launch across several codebases (Python, C#, TypeScript), owning REST APIs, a card + crypto payments system, serverless GPU workers, and the CI/CD behind it.

SKILLS

Languages: C#, Python, TypeScript, SQL, Bash, HCL (Terraform), C++, C

Backend & APIs: FastAPI, Pydantic v2, AWS Lambda Powertools, uvicorn, ASP.NET, REST API design, JSON-Schema structured output, JWT (PyJWT) / API-key (`x-api-key`) auth

Async & Orchestration: Inngest (durable step functions — `step.run` / `step.wait_for_event`, event-driven webhook resume, timeout handling), n8n, AWS EventBridge Scheduler

Cloud & Infrastructure: AWS (Lambda, S3, API Gateway HTTP API v2, EventBridge, App Runner, ECR, VPC + private subnets, IAM, Lambda Function URLs), Terraform (multi-environment modules), Docker, RunPod (serverless GPU), self-hosted VPS w/ blue-green deploys, nginx (reverse proxy / upstream cutover), systemd, Tailscale (WireGuard mesh VPN), GitHub Actions (CI/CD), Poetry, pytest / unittest

Databases: PostgreSQL (RDS), psycopg2 (RealDictCursor, parameterized queries, stored procedures), connection pooling (node-postgres `pg.Pool`)

AI / ML & Services: fal.ai (nano-banana-2, Flux, finetuned LoRA), Kling 2.6, WAN 2.1 LoRA, Segmind, Runway, Grok (xAI), Gemini 2.0/3.1 Pro, OpenAI, OpenRouter, Portkey (AI gateway / observability), promptfoo (evals, LLM-as-judge), audio-separator (UVR / MelBand Roformer), LaMa / OpenCV inpainting

Media & Web3: Cloudinary, Creatomate, Rendi (FFmpeg-as-a-service), Cloudflare Images / KV, FFmpeg, Stripe, Helio, Solana / Helius (on-chain verification)

Bots: discord.js v14, Telegraf (Telegram)

Game Development: Unity (gameplay, particles, physics, URP, WebGL), VR/AR (XR Toolkit, MRTK3, JioMRTK, Oculus SDK), Unreal Engine, DOTween, LeanTween, Photon, MoreMountains Feedbacks

WORK EXPERIENCE

Sandwatch, Berlin, Germany (Remote) **FEB 2025 – PRESENT**

Software / AI Engineer

Shipped an AI video-generation platform from MVP to public launch (190+ PRs on the core engine, hundreds of merged changes), working across the full stack — Python API backend, C# generation engine, cloud infrastructure, TypeScript bots, and GPU workers.

AI agent pipeline (durable, event-driven)

- Built a durable, event-driven AI generation pipeline — FastAPI with zero-downtime blue-green releases, orchestrated with Inngest durable step functions — chaining LLM calls, submitting async work to external AI providers, and suspending/resuming execution on webhook callback with request-ID matching and timeout handling.
- Designed the end-to-end generation as a ~20-step durable pipeline spanning script, voice/TTS, character, and multi-stage visual generation, with webhook-resumed provider calls and strict JSON-Schema structured output at every LLM step.
- Implemented a self-discovering service-URL pattern for webhook callbacks, with a local tunnel fallback for development.
- Built a custom OpenRouter-via-Portkey LLM client (pure urllib, no SDK) with structured output, reasoning-effort controls, distributed trace IDs, and metadata tagging for observability.

Prompt evaluation & quality (LLM-as-judge)

- Built a promptfoo-based eval framework with a custom provider (delegating to the Portkey/OpenRouter client) and a multimodal LLM-as-judge that grades generated `.mp4` videos — base64-encoded as a `video_url` content part to Gemini — returning structured pass/score/reason verdicts via JSON-Schema `response_format` (strict mode).
- Wrote a deterministic rule-based visual critique pre-filter (bans generic/overused descriptors) that runs before the LLM critique to cut cost and tighten quality.
- Created a CLI eval runner over registered pipeline steps with HTML reports and a `MOCK_LLM` mode for offline development.
- Codified output quality into 11 LLM-as-judge eval modules (topic, voiceover, pacing, structure, blueprint, template-extraction) within an automated suite of ~198 tests across 33 modules, gated in CI — the GitHub Actions unit-test workflow runs the full suite on every pull request to `main/dev`, blocking regressions before merge.

Creator-platform REST API (Python / AWS Lambda)

- Shipped 30+ REST endpoints on AWS Lambda (Lambda Powertools router, Pydantic v2 request/response models, psycopg2 + `RealDictCursor` over PostgreSQL, parameterized queries) covering agents, video structures, characters, video presets, generations, caption styles, topics, and home-screen aggregation.
- Built character preview generation via fal.ai async webhook integration with credit deduction and status polling.
- Built a central library endpoint aggregating a user's generated videos with filters (agent, preset, character, duration), including structured voiceover in results.
- Added voice catalog filter & search, preset sharing from the discovery page (single home-entry endpoint), and the ability to remove failed generations from a user's feed.
- Added fallback handling for scene-prompt and image-generation failures so partial pipeline failures degrade gracefully.

Monetization: credits & payments

- Built the billing/credit system end to end — atomic charge-on-create in a single DB transaction (charge committed with the row insert) with the charged cost persisted for exact, source-aware refunds via PostgreSQL stored procedures.
- Integrated card (Stripe) and crypto checkout for the credit store, including on-chain payment verification — parsing and persisting sender-wallet data alongside card and blockchain orders.
- Delivered reconciled, dispute-free billing with no accounting discrepancies, backed by a thorough unit + integration test suite gated in CI before deploy.

Voice platform

- Migrated the voice catalog to a new TTS/cloning provider and built the cloning flow — hiding internal provider IDs from the client and handling name collisions cleanly.
- Scaled the catalog to a 600-voice multilingual library (50 voices × 12 languages) via a scrape → clean → seed pipeline.
- Added social-source voice-clone ingestion with lipsync support.

Multi-provider resilience (video generation)

- Built a multi-provider fallback chain across video-generation vendors with terminal-failure detection, provider reordering, a post-generation upscaling step, and time-window gating.
- Adding a fallback provider cut the video-generation failure rate by more than half and reduced average generation time, hardened by an automated fallback test suite covering every provider path.

Serverless GPU workers (CV / audio ML)

- Built a RunPod serverless GPU worker for AI vocal removal — an ensemble of specialist source-separation models via the `audio-separator` library (UVR-MDX-NET ONNX + MelBand Roformer CKPT) baked into the CUDA Docker image at build time, NumPy int32-PCM stem-averaging of the instrumental outputs, and S3 presigned-URL delivery (boto3).
- Built a second RunPod GPU worker for automated video inpainting / object removal — detects the target region via OpenCV Canny edge-density × temporal-stability scoring across sampled frames, builds a mask, and inpaints with

OpenCV TELEA (CPU) or the LaMa model (CUDA/GPU), processing frames in batches and re-muxing the original audio track with ffmpeg.
- Baked model weights into the Docker image at build time to eliminate cold-start model download and tuned GPU frame batching (128–256 frames/batch) for throughput.

Autonomous C# AI video engine (sole developer, ~190 PRs)
- Built a node-based content-generation engine on .NET / C# (custom DAG execution engine) orchestrating text-to-image and image-to-video models, LoRA fine-tunes for character consistency, LLM story generation, cloud video splicing/merging, and voice synthesis into fully automated long-form social content.
- Shipped full long-form story pipelines (LLM structured-output plotting), multi-reference image generation for character consistency, 16:9 reframing, automated image-quality assessment + upscaling, and an SRT-based subtitle system.
- Integrated per-scene voiceovers, audio fade transitions, and cloud-based scene assembly/merging; continuously upgraded the image/video model stack as better models shipped.

Infrastructure as Code (Terraform / AWS)
- Managed all AWS infrastructure with Terraform across isolated prod and dev environments — serverless functions, storage, event scheduling, private networking, and IAM — enabling reproducible deploys and independent release cycles.
- Provisioned service modules end to end (private subnets, service URLs, sensitive secret variables) and wired secrets through the CI/CD pipeline.

CI/CD & deployment
- Owned the deployment pipeline end to end — migrated the service off managed functions onto self-hosted infrastructure, writing the server provisioning script and GitHub Actions workflows that build, ship secrets securely, and deploy remotely, with separate prod/dev pipelines and a concurrency lock.
- Implemented zero-downtime blue-green releases — deploy to the idle instance, health-gated cutover behind a reverse proxy, then drain the old instance before retiring it; services run under process supervision.
- Diagnosed and fixed unreachable-runner deploy failures — added network-path diagnostics to isolate the issue, then routed deploys over a private mesh VPN (Tailscale/WireGuard), fixing reachability and hardening security by closing SSH to the public internet.
- Took deploys from intermittently failing to consistently reliable, zero-downtime releases; managed secrets for many third-party providers, injected securely at deploy time.
- Authored the pytest CI workflow for the Python backend (runs the shared, payment, and core test suites via Poetry on pull requests) and fixed deploy change-detection so shared-library changes correctly redeploy dependent functions.

Discord / Telegram bots (TypeScript)
- Built a TypeScript Discord bot (discord.js v14) with agent-selection slash commands, S3 presigned-URL video delivery (`@aws-sdk/s3-request-presigner`), and a PostgreSQL API client over a `pg.Pool`; one video delivered per message with hyperlink formatting.
- Implemented Solana on-chain payment verification (Helius RPC) alongside Stripe for dual-payment agency order flows.
- Added a process-local `Map`-based TTL cache (5-minute eviction) on hot read paths (user profiles, agent lists) to cut redundant PostgreSQL round-trips on frequent Discord commands.

Internal tooling & content ops
- Empowered non-technical staff to publish landing-page content themselves — built a Python CLI sync tool that turns showcase updates into a drop-a-folder-and-run workflow (no code, no developer, no redeploy), removing engineering from the content-publishing loop that previously required a code edit and site redeploy per change.
- Under the hood: scans a folder structure, parses per-video metadata, uploads avatars to Cloudflare Images, and pushes the showcase JSON to Cloudflare KV so the live site picks up changes without a deploy.
- Implemented SHA-256 content-based image deduplication (paginating existing CDN images and matching by content hash, not filename) so re-syncs upload only new/changed avatars — zero redundant CDN uploads.

Sacher Solutions, Mediaș, Romania (Remote) **JUL 2024 – PRESENT**

Unity / .NET Developer

- Built a mobile/WebGL endless runner for a consumer-brand marketing campaign (agency contract) — sole developer, 93 commits from scratch: obstacle pooling, progressive difficulty, full UI/menu state machine, audio sequencing and haptics, game-feel feedback (MoreMountains), and a giveaway form wired to a backend REST API.
- Built an interactive Unity video-station app — animated tile grid, video player with full controls (play/pause/mute, next/previous), popup UI with overlay, fade-in transitions, and per-station detail views.
- Reduced frame time and peak memory across shipped titles through Unity Profiler–guided optimization — object-pool tuning, reduced per-frame GC allocations, cache-friendly data layouts on hot paths.
- Decoupled gameplay, UI, and audio using service-locator and dependency-injection patterns, keeping modules independently testable and shippable across products.
- Drove character and UI animation with LeanTween/DOTween across titles, cutting animation-code duplication and holding frame budgets on target devices.

Freelance, Lagos, Nigeria **JAN 2022 – JUN 2024**

Unity / .NET Developer

- Developed immersive VR and AR projects (C#, Unity, XR Toolkit, Oculus Integration SDK, MRTK3, JioMRTK) targeting Quest 2, Quest 3, and Jio MR glasses.
- Built multiplayer WebGL/mobile games using Photon.
- Created a simulation with a dynamic, intelligent agent using A* pathfinding in Unity/C#.

BimiBoo Kids, California, US (Remote) **AUG 2022 – NOV 2022**

Unity / .NET Developer

- Built 6 mobile hypercasual kids' games with Unity and C#.
- Implemented performant, child-friendly touch interactions and animations (DOTween).

LEADERSHIP AND VOLUNTEERING

Deputy Training Lead — Engineering Career Expo **SEP 2024 – PRESENT**

Partnership Team Member — Google Developer Student Club UNILAG **OCT 2023 – NOV 2024**

Game Development Tutor — Engineering Career Expo **DEC 2021 – JUL 2024**

Communications Team Member — Shell Eco-Marathon UNILAG **OCT 2020 – AUG 2021**

EDUCATION

University of Lagos, Akoka, Lagos **OCT 2019 – SEP 2025**

Bachelor of Science (B.Sc.) Civil Engineering — CGPA: 4.33 / 5