Oladele Usman

uoladele99@gmail.com · +234 706 343 2968 · github.com/codetesla51 · devuthman.vercel.app · dev.to/uthman_dev · Ilorin, Nigeria

Go backend engineer who builds from scratch to understand before abstracting — a scripting language, a distributed job queue, a rate limiter, and an HTTP server from raw TCP sockets. Strong on reliability patterns and correctness under concurrency.

TECHNICAL SKILLS

Language Go (concurrency, TCP networking, interpreter design, distributed systems primitives)
Backend REST APIs, HTTP/1.1, TCP/IP, TLS/SSL, SSE, Webhooks, JWT
Databases PostgreSQL, Redis
Tools Docker, Git, Linux, Prometheus, Grafana

EXPERIENCE

University of Ilorin, COMIST (Website Unit)

Industrial Training · Ilorin, Nigeria · Apr 2026 – Present

— Maintained the university's official web presence using WordPress — updating content, managing pages, and ensuring accuracy across student-facing sections.

— Resolved student email access issues, coordinating with the IT team to restore accounts and fix authentication failures.

Odelade Family Ledger (Ajo Portal) — Freelance Contract

Go · PostgreSQL · Redis · Paystack · Supabase · Apr 2026 · Codebase private

— Full-stack: Go REST API backend deployed on Render, HTML/CSS/JS frontend on Vercel, serving 20+ active users. Two independent JWT auth systems (member + admin) via Seal ensure tokens cannot cross privilege boundaries.

— Dual-process architecture: HTTP server and job worker run as separate deployable binaries — care fund approvals process asynchronously via kyu with automatic retries, and a daily goroutine detects overdue contributors and triggers in-app notifications without manual intervention.

— Designed for correctness under concurrent writes: double-entry ledger with SERIALIZABLE transactions and row-level locking on pool transfers eliminates race conditions. Ships with a full audit log, Supabase receipt storage, and CSV transaction export.

PROJECTS

Logos — Scripting Language github · logos-lang.vercel.app

Go · Pratt Parser · Tree-walking Interpreter · Apr 2026

— Built because existing scripting languages felt either too heavy or too unreadable — wanted something lightweight, batteries-included, and clean to write.

— Dynamically-typed language built from scratch: Pratt parser, tree-walking interpreter, closures, pipe operator, and built-in concurrency via a spawn keyword.

— Result-based error handling via a try expression — failures propagate up the call stack as structured result tables rather than panicking, giving scripts predictable failure behaviour.

— Exposes a Go API for embedding Logos into Go programs — register native Go functions callable from scripts and call Logos functions directly from Go.

kyu — Distributed Job Queue github · kyu-job-queue.vercel.app

Go · PostgreSQL · Redis · Prometheus · Grafana · Mar 2026 · Open source · 1 external contributor

— Built to learn distributed task processing, retries, scheduling, and worker crash recovery by implementing a job queue from scratch.

— Dual-store architecture: Redis handles fast dispatch via a sorted set priority queue; PostgreSQL is the source of truth. Jobs survive a full Redis wipe because no critical state lives only in the cache.

— Three components run concurrently: a worker pool, a scheduler that promotes jobs when their time arrives, and a stale reaper that detects crashed workers and requeues them automatically. Exponential backoff on retries.

— Benchmarked at 52ns/op with 0 allocations on job registration, 950ns/op on execution (Go benchmark tooling). Ships with a Prometheus metrics endpoint and a pre-built Grafana dashboard covering queue depth, throughput, and failure rates by job type.

limitz — Rate Limiting Library github

Go · Redis · PostgreSQL · Feb 2026

— Five algorithms (Token Bucket, Fixed Window, Sliding Window Log/Counter, Leaky Bucket) behind a single common interface — swap algorithms without changing application code.

— Pluggable storage with automatic Redis → PostgreSQL failover keeps rate limiting active even when cache goes down. Sub-millisecond latency on most algorithms.

— Context-aware throughout: cancellation and deadlines propagate into every storage operation, so a timed-out request aborts the rate limit check cleanly rather than hanging.

seal — JWT Authentication Library github

Go · Redis · PostgreSQL · MySQL · SQLite · Jan 2026

— Single-use refresh token rotation: each refresh invalidates the previous token, so a stolen token can only be used once before it becomes worthless. Tokens are SHA-256 hashed before storage — the raw value never touches the database.

— Optional Redis blocklist for immediate access token revocation on logout. RevokeAll forces re-login across every active session simultaneously.

— Four swappable storage backends and drop-in HTTP middleware for route protection.

Raw-HTTP — HTTP/1.1 Server from TCP Sockets github

Go · TCP/IP · TLS · Aug 2025

— Built to understand what net/http abstracts away — parsing, keep-alive, chunked encoding, and TLS at the socket level.

— HTTP/1.1 built on raw TCP: custom request parser, path parameters, keep-alive, chunked encoding, TLS, static file serving with MIME detection, and graceful shutdown on SIGINT/SIGTERM.

— Benchmarked at 11,042 req/s under 500 concurrent connections (ApacheBench), 45.3ms latency — up from 250 req/s before replacing per-request allocations with three sync.Pool buffer pools.

— Every connection handler wrapped with panic recovery so a handler crash returns a 500 without bringing down the server. Path traversal attacks on static file routes are blocked explicitly.

go-git — Git Implementation in Pure Go github

Go · SHA-256 · zlib · Oct 2025

— Implemented Git's core object model in pure Go without external libraries: content-addressable SHA-256 storage, zlib-compressed blobs, trees, commits, and a three-tree staging architecture.

— Implemented correct tree construction, object hashing, binary wire format, and staging — covering the full lifecycle from working directory to committed object.