**G8S LPG IDO Platform — Documentation Report**

This report consolidates company/financial context, token design, calculations, AI prompting approach, and the end‑to‑end build process for the current Supabase‑native MVP.

## 1) Company Overview

- \*\*Vision\*\*: Accelerate clean‑energy adoption in Africa by tokenizing LPG distribution and financing with transparent, on‑chain rails.

- \*\*Mission\*\*: Make LPG access affordable and reliable by connecting retail demand, logistics, and financing through a tokenized ecosystem (G8S).

- \*\*Products/Services\*\*:

- G8S token and IDO portal for on‑chain fundraising.

- Investor UI to purchase G8S with PUSD (Sepolia).

- Admin analytics via frontend and Supabase Studio.

- \*\*Target Audience\*\*: Retail/institutional investors aligned with clean energy; LPG distributors; early Web3 adopters across Africa.

- \*\*Unique Value Proposition\*\*:

- Verifiable, auditable token sale flows on‑chain.

- Simple UX with stablecoin pricing and NGN equivalents.

- Modern stack (Next.js, Supabase, Railway) for speed, security, and scale.

---

## 2) Business Model

- \*\*Primary Revenue\*\*: Token sale proceeds (IDO) used for treasury, operations, expansion.

- \*\*Future Revenue\*\* (optional): Platform fees on logistics/marketplace integrations, partner programs, loyalty.

- \*\*Sustainability\*\*: Treasury allocation for operations; community incentives to grow usage; prudent reserves.

---

## 3) Valuation (Assumptions)

- Early MVP with deployed contracts and functioning sale flow on Sepolia.

- Benchmarks for clean‑energy Web3 raises: $0.03–$0.10 per token common for early rounds.

- Two valuation scenarios based on token price per G8S (PUSD):

- \*\*Scenario A (Baseline)\*\*: $0.05 → Fully Diluted Valuation (FDV) = 1,000,000,000 × $0.05 = $50,000,000 (≈ ₦75,000,000,000 at ₦1,500/$)

- \*\*Scenario B (Aggressive)\*\*: $0.10 → FDV = $100,000,000 (≈ ₦150,000,000,000)

- Rationale: The schema’s seeded `price\_per\_token` ≈ $0.05 anchors Scenario A as a consistent baseline.

---

## 4) Fundraising Target (NGN)

- \*\*Target\*\*: ₦22,500,000,000 (₦22.5B) in Scenario A.

- Derived by selling 300,000,000 tokens (30% of 1B) at ₦75 each (≈ $0.05 × 1,500 NGN/USD).

---

## 5) Token Design

- \*\*Name / Symbol\*\*: G8S Coin / G8S

- \*\*Total Supply\*\*: 1,000,000,000 G8S (on‑chain)

- \*\*IDO Allocation\*\*: 30% = 300,000,000 G8S (funded to IDO in deployment)

- \*\*Tokenomics (distribution)\*\*:

- IDO/Public Sale: 30% (300,000,000)

- Treasury/Operations: 30% (300,000,000)

- Team & Advisors (vesting): 20% (200,000,000)

- Strategic Reserves/Partnerships: 15% (150,000,000)

- Community Incentives/Liquidity: 5% (50,000,000)

- \*\*Vesting/Lockups (recommended)\*\*:

- Team & Advisors: 12–24 months vesting, 6‑month cliff.

- Treasury/Reserves: multisig, disclosed schedule.

- Incentives: programmatic release.

Why this is correct:

- Contracts and deployment set total supply = 1B and IDO allocation = 300M.

- The rest of the buckets are realistic for an early‑stage ecosystem and sum to 100%.

---

## 6) Raise Calculation (Formulas & Results)

Let:

- \( S \) = tokens for sale in IDO = 300,000,000

- \( P\_{usd} \) = price per G8S in PUSD (USD)

- \( R \) = NGN per USD (use 1,500 for baseline)

Per‑token NGN price:

\[ P\_{ngn} = P\_{usd} \times R \]

Total raise:

\[ \text{Raise}\_{usd} = S \times P\_{usd} \]

\[ \text{Raise}\_{ngn} = S \times P\_{ngn} = S \times P\_{usd} \times R \]

- \*\*Scenario A (baseline)\*\*: \( P\_{usd} = 0.05, R = 1{,}500 \)

- \( P\_{ngn} = 75 \)

- \( \text{Raise}\_{usd} = 300{,}000{,}000 \times 0.05 = 15{,}000{,}000 \)

- \( \text{Raise}\_{ngn} = 300{,}000{,}000 \times 75 = ₦22{,}500{,}000{,}000 \)

- \*\*Scenario B (higher price)\*\*: \( P\_{usd} = 0.10 \)

- \( P\_{ngn} = 150 \)

- \( \text{Raise}\_{usd} = 30{,}000{,}000 \); \( \text{Raise}\_{ngn} = ₦45{,}000{,}000{,}000 \)

Note: If you prefer a target token price in NGN (e.g., ₦666.67), compute \( P\_{usd} = 666.67 / 1{,}500 \approx 0.444 \) then apply the same formulas.

---

## 7) Prompts Used with AI Agents

- \*\*Frontend Agent\*\*

- Identity: Senior React/Next.js engineer optimizing RainbowKit/Wagmi/Viem wallet flows.

- Task: Implement Approve + Buy with async `writeContractAsync`, `useWaitForTransactionReceipt`, Sepolia chain switching, and decimals‑safe `parseUnits`. Show clear pending/confirmed states.

- Constraints: Don’t break SSR; keep styling; use env vars; avoid unrelated edits.

- \*\*Backend Agent\*\*

- Identity: Express + Supabase engineer on Railway.

- Task: Defer Supabase client creation, trim envs; add startup checks; disable Mongo via `USE\_MONGO=false`; support HTTP provider (skip event listeners) and optional WS provider.

- Constraints: Don’t break API; keep health check passing; never leak secrets.

- \*\*Smart Contract Agent (advisory)\*\*

- Identity: Solidity reviewer.

- Task: Validate price math (18 decimals), cap checks, reentrancy guards, Pausable; verify transfer success logic.

- Constraints: Keep interfaces/constructor invariants.

Each prompt always stated identity, task, and constraints to reduce ambiguity and ensure safe, incremental edits.

### 7.1 Augmented From This Project’s Chat History

Below are distilled prompt patterns actually used during our collaboration, organized by domain, with the intent, constraints, and acceptance criteria that made them effective.

- \*\*Frontend (Wallet + UX)\*\*

- Intent: “My ‘Approve’ button doesn’t respond on Vercel; make it reliable.”

- Constraints: Keep UI design, don’t break SSR, use env addresses, support Sepolia, show pending/confirmed states.

- Resolution prompts:

- “Use `writeContractAsync` + `useWaitForTransactionReceipt` and track tx hashes.”

- “Switch to Sepolia before sending tx; use `parseUnits` with ERC20 `decimals()`.”

- “If HTTP RPC is slow, show ‘Waiting for confirmation…’ until receipt.”

- \*\*Backend (Supabase‑native, Railway)\*\*

- Intent: “Railway builds but /health fails; app crashes with ‘Invalid supabaseUrl’.”

- Constraints: No secrets in code; keep server alive on partial failures; pass healthcheck.

- Resolution prompts:

- “Trim envs; defer `createClient` until constructor; guard when keys are missing.”

- “Add startup check to assert required envs; log which are missing.”

- “Prefer WS provider when available; if HTTP only, skip event subscriptions.”

- “Add `USE\_MONGO=false` and guard all Mongoose calls + cron tasks.”

- \*\*Infra & CORS\*\*

- Intent: “Healthcheck failing; CORS; which envs belong where?”

- Constraints: Keep Vercel/Railway standard; do not change service roots.

- Resolution prompts:

- “Frontend set `NEXT\_PUBLIC\_API\_URL` to Railway; backend set `FRONTEND\_URL` to Vercel domain.”

- “Expose `/health`; verify logs; redeploy after env changes.”

- \*\*Admin Access\*\*

- Intent: “/admin returns 404/blocked; how to make an admin?”

- Constraints: Use Supabase tables/roles.

- Resolution prompts:

- “Add `role` column; set `users.role='admin'`; sign in then visit /admin.”

- \*\*Reporting & Tokenomics\*\*

- Intent: “Produce PRD/testcases/README + a fundraising report, ensure tokenomics correct.”

- Constraints: Match on‑chain supply (1B) and sale (300M); make raise math explicit; support NGN.

- Resolution prompts:

- “Anchor baseline price to schema ($0.05); compute raise for 300M; show variants; explain adjustments.”

### 7.2 Effective Prompt Templates Used

- Identity: “You are a senior X engineer (React/Express/Solidity)…”

- Task: “Implement/fix Y with Z constraints…”

- Constraints: “Do not change unrelated files; keep SSR; no secret leakage; pass healthcheck…”

- Acceptance: “Wallet pops promptly, shows pending/confirmed; /health=200; logs show all required envs.”

### 7.3 Anti‑Patterns Avoided

- Initializing Supabase clients at import time (fixed by deferring init).

- Subscribing to events over HTTP providers (guarded + optional WS path).

- Mixing Mongo writes while disabling Mongo (guarded via `USE\_MONGO`).

- Using `parseEther` for ERC‑20 amounts with non‑18 decimals (switched to `parseUnits(decimals)`).

### 7.4 Prompts Grounded in `cursor\_create\_product\_requirements\_and.md`

That file guided multi‑doc outputs (PRD, test plan/cases, deployment) and acceptance criteria. Effective prompts derived from it:

- Product Requirements Synthesis Agent

- Identity: “You are a product strategist turning stakeholder notes into a PRD with scope, roles, flows, NFRs.”

- Task: “Consolidate and structure requirements into sections (overview, personas, flows, APIs, data model, NFRs). Remove contradictions.”

- Constraints: “No speculative features; align with current stack and contracts; keep it implementation‑ready.”

- Acceptance: “Outputs `PRODUCT\_REQUIREMENTS.md` with clear sections and unambiguous success criteria.”

- QA Test Planner Agent

- Identity: “You are a QA lead creating a practical test plan and test cases.”

- Task: “Produce a compact plan (scope, approach, environments, entry/exit) and a separate detailed test cases matrix (positive/negative/edge).”

- Constraints: “Prioritize auth, wallet flows, contracts, admin roles, env/CORS; include deployment checks.”

- Acceptance: “Outputs `TEST\_CASES.md` with IDs, steps, and expected results; covers env/health, auth, wallet, admin, security, UX.”

- Deployment & Ops Agent

- Identity: “You are a DevOps engineer for Vercel/Railway/Supabase.”

- Task: “List exact env variables per platform; health endpoints; WS vs HTTP provider guidance; troubleshooting.”

- Constraints: “Do not suggest changing service roots; avoid secrets in code; prefer health‑first, fail‑safe startup.”

- Acceptance: “Outputs a deployment guide and README that work copy‑paste.”

- Documentation Consolidation Agent

- Identity: “You are a technical writer harmonizing docs.”

- Task: “Create a Supabase‑native README, and a fundraising/process report with tokenomics and NGN math.”

- Constraints: “Reflect on‑chain facts (1B supply, 300M sale); explicit formulas; keep docs concise and actionable.”

- Acceptance: “`README\_SUPABASE.md` and this report contain accurate endpoints, repo link, and cohesive tokenomics.”

---

## 8) Process Report

- \*\*Prompt Structure\*\*: Defined agent identity, task, constraints; requested minimal diff changes; avoided unrelated churn.

- \*\*Debugging\*\*:

- Supabase init: removed import‑time `createClient`, trimmed envs; added guard logs.

- Ethers subscriptions: added `SEPOLIA\_WS\_URL` path; on HTTP, skip `.on()` to prevent crashes.

- Mongo timeouts: introduced `USE\_MONGO=false`; guarded model calls and cron jobs.

- Frontend UX: switched to `writeContractAsync`, tracked `approvalHash`/`purchaseHash`, displayed `Waiting for confirmation…` states; added chain switch and decimals‑safe `parseUnits`.

- \*\*Validation\*\*:

- Railway healthcheck returns 200; logs show all required envs present.

- JWT protects private routes; role gating for admin endpoints.

- No secrets in code; env‑only configuration.

- \*\*Integration\*\*:

- Frontend uses Wagmi/Viem + RainbowKit; env‑driven contract addresses.

- Backend uses Supabase Auth and `users` table for profiles; JWT payload stores Supabase user id; CORS aligns with Vercel domain.

- Contracts provide 1B supply and 300M sale cap; IDO math is deterministic and guarded.

---

## Appendix — Implementation Notes

- \*\*Contracts\*\*: `G8SToken` mints 1B to deployer; `G8SIDO` receives 300M allocation and enforces price, cap, and Pausable + nonReentrant protections.

- \*\*Frontend\*\*: `IDOPurchase.tsx` handles approve/purchase, chain switching, receipt tracking, and validation (balance/allowance/paused).

- \*\*Backend\*\*: Supabase‑native auth (register/login/me) with JWT; robust Supabase init; optional WS provider for events; Mongo disabled by default.

- \*\*Admin\*\*: Set `users.role='admin'` in Supabase to access `/admin`.

---

## Summary

- Tokenomics are harmonized to the deployed code: 1B supply, 30% IDO (300M), price baseline $0.05 (schema), target raise ≈ ₦22.5B at ₦1,500/$.

- All calculations are explicit and re‑computable for any chosen price/FX.

- The stack is now Supabase‑native, production‑deployable, and resilient to typical infra issues (RPC latency, missing envs, event subscription limits).

---

## Deployed Endpoints

- Frontend (Vercel): https://g8s-lpg.vercel.app/

- Backend (Railway): https://g8s-lpg-api.up.railway.app

- GitHub Repository: https://github.com/G8Supremeo/G8S-LPG-IDO-Platform.git