EzyGas Replit Production Readiness Scorecard

Full Audit Synthesis — April 10, 2026

Executive Summary

The codebase is **far more mature than the hosting environment suggests**. This is not a Replit prototype with "gas-themed labels" — it is a genuinely architected LPG asset platform with a proper domain model, state machines, ledger-based finance, and multi-role auth. The business logic layer is 75% production-grade.

But it is sitting on a foundation with **four critical defects that would individually justify halting a national launch**, and a constellation of HIGH-severity gaps that compound under load. The blocking issues are not architectural — they are discipline issues: hardcoded credentials, unrotated secrets, missing idempotency on money, and a Personal Access Token sitting in plaintext in `.git/config`.

Verdict in one line: The code is worth migrating. The current deployment is not worth launching on.

Readiness Scorecard (0–5 scale)

#	Domain	Score	Evidence
1	Runtime & Deployment Config	**4/5**	Autoscale deployment configured, Node 20 pinned via Nix, ports bound correctly. Missing: `engines` field in package.json, no alwaysOn for background workers.
2	Data Architecture	**4/5**	External Neon Postgres, Drizzle ORM, proper migrations, R2 for objects, Upstash for cache. Real architecture.
3	Data Durability & Backup	**1/5**	**Zero application-level backup strategy.** Sole protection is Neon's platform default, undocumented and unverified. No DR runbook.

4	Secrets Hygiene	**1/5**	**CRITICAL:** plaintext staff passwords in source + git history, GitHub PAT in `.git/config`, VAPID private key in unencrypted shared env, ~15 missing production secrets.
5	Authentication & Authorization	**3/5**	Three auth flows implemented correctly in structure, bcrypt used, but hardcoded seed credentials + rate limiter path mismatch + one admin route with consumer auth.
6	Payment Integrity	**2/5**	HMAC verification ✅, server-side amounts ✅, **but no webhook idempotency and no indexed Paymob transaction ID**. Duplicate ledger entries on retry are guaranteed at scale.
7	Dependency Health	**2/5**	16 vulnerabilities (1 critical, 9 high), drizzle-orm SQL injection in the primary ORM, axios SSRF. 121 of 123 packages floating.
8	Observability	**2/5**	Sentry DSN-gated but routes bypass it (inline `res.status(500)`), no `unhandledRejection` handler, ~15 silent catches including `autoDispatchOrder`.
9	Source Control Discipline	**1/5**	**PAT embedded in `.git/config`**, Cloudflare token screenshot committed, no CI/CD, no branch strategy, `.gitignore` missing `.env`/`*.sqlite`/`logs`.
10	Business Domain Integrity	**4/5**	Cylinder lifecycle, order state machine, ledger, pricing control, inventory — all real. Main gaps: distributor tier hierarchy and penalty matrix absent from code.

Weighted overall readiness: 2.4 / 5 — NOT production-ready for national launch. Blocking defects in 4 domains.

Critical Defect List (Ranked by Blast Radius)

Tier 0 — Launch Blockers (Fix Before Any Public Traffic)

D0.1 — GitHub PAT in `.git/config` A `ghp_...` token is sitting in plaintext inside the Repl filesystem. Anyone with filesystem access — including any AI agent, any shared session, any backup — has full write access to your GitHub repo. This is the single most urgent finding in the entire audit because it is trivially exploitable *right now*. **Fix (today):** Revoke the PAT on GitHub. Re-add the remote without embedded credentials (use SSH key or Replit's GitHub integration).

D0.2 — Hardcoded plaintext passwords in `server/storage.ts:1547–1549` `admin/admin123` plus two real staff accounts (`melzayat@ezy-gas.com`, `msoubhy@ezy-gas.com`) with real passwords, committed to git history. This seed runs on every boot. The git history cannot be un-leaked. **Fix (today):** (1) Change both staff passwords in production immediately. (2) Remove the hardcoded array; rely on `ADMIN_PASSWORD` env lookup only. (3) Delete the `admin/admin123` default or protect it behind `NODE_ENV === 'development'`. (4) Audit *every other system* those two passwords might have been reused on.

D0.3 — Paymob webhook has no idempotency and no indexed transaction ID Paymob retries webhooks. Every retry writes a new `eventsLedger` row for the same real payment. There is no way to deduplicate because the Paymob ID is buried in an unindexed JSONB payload. At launch volume, your financial records will drift from Paymob's within the first week. **Fix (this week):** Add `paymob_transaction_id VARCHAR UNIQUE` column to `orders`. Check `order.paymentStatus !== 'paid'` before writing. Upsert on the unique constraint.

D0.4 — Cloudflare API token screenshot committed in commit `f1db226` If the screenshot shows actual token values, those tokens are permanently in git history. Cloudflare tokens typically have R2, DNS, and worker permissions — this is potentially full infrastructure compromise. **Fix (today):** Inspect the image in the commit. If tokens are visible, rotate every Cloudflare API token immediately, and decide whether to scrub git history (`git filter-repo`) or accept the leak and rotate exposed credentials permanently.

Tier 1 — High-Severity (Fix Before Phase 1 City Launch)

D1.1 — Admin login rate limiter path mismatch. Limiter registered on `/api/admin/login` but real endpoint is `/api/admin-auth/login`. Admin login is effectively unrate-limited. Fix is a one-line path correction.

D1.2 — `/api/admin/cylinders/:id/sensor` uses consumer auth. Any logged-in customer can modify sensor settings on any cylinder in the fleet. Swap `isAuthenticated` for `isAdminAuthenticated`.

D1.3 — drizzle-orm SQL injection (GHSA-gpj5-g38j-94v9). Your primary ORM has a known SQL injection vulnerability. Upgrade to ≥0.45.2.

D1.4 — axios SSRF (critical CVE). Transitive dep, needs an `overrides` entry in package.json forcing axios ≥1.15.0.

D1.5 — Admin logout doesn't destroy session. Session cookie remains valid for 7 days after logout. Add `req.session.destroy()`.

D1.6 — VAPID private key stored as unencrypted shared env var. Move it to encrypted Secrets panel.

D1.7 — Missing production secrets (Paymob, R2, SMSMisr, SMTP_PASS, Upstash, Slack). According to the audit, these are referenced in code but not set. Either they are set under different names (and the audit missed them) or core platform functions are silently failing. Resolve this ambiguity by printing a secret-presence table in a startup validator.

D1.8 — `autoDispatchOrder().catch(() => {})`. Driver assignment failures are completely invisible. An order can silently remain unassigned with no alert. Log the error and push to Sentry at minimum.

D1.9 — No `unhandledRejection` / `uncaughtException` handlers. One uncaught promise rejection in a background timer can crash the process with zero trace.

D1.10 — No automated backup. Neon's platform backup is not sufficient for a regulated LPG platform. Implement daily `pg_dump` to R2 with tested restore.

D1.11 — No Paymob reconciliation job. If a webhook ever fails silently, money is collected but never acknowledged. Daily reconciliation against Paymob's transaction API is non-negotiable for a cash platform.

D1.12 — `drivers.nationalId` stored in plaintext. Government IDs in unencrypted columns. Egyptian data protection law exposure. Encrypt at application layer or use column-level encryption.

Tier 2 — Medium-Severity (Fix Before Nationwide Expansion)

- In-memory OTP store (loses state on restart)
- `orders.otpCode` stored in plaintext (should be hashed)
- `price` and `paymentStatus` accepted in `createOrderSchema` pick (landmine for future devs)
- Hardcoded Paymob iframe ID fallback `405966` (may be sandbox — verify)
- Routes bypass Sentry by using inline `res.status(500)` instead of `next(err)`
- No CI/CD pipeline — zero automated gates on push
- No staging/production branch separation
- Distributor tier hierarchy (Entry/C/B/A Elite) missing from schema
- Penalty matrix not coded — currently spreadsheet-based
- `.gitignore` missing `.env`, `*.sqlite`, `logs/`, `uploads/`, `secrets/`
- No external uptime monitoring
- No structured logging — stdout soup under load

30-Day Remediation Plan

Week 1 — Stop the Bleeding (All Tier 0 + Critical Credential Hygiene)

Day	Action	Owner	Verification
1	Revoke GitHub PAT, re-add remote without embedded credentials	Tech lead	`git remote -v` shows no token
1	Inspect commit `f1db226` for visible Cloudflare tokens; rotate if exposed	Tech lead	New tokens in place, old revoked
1	Change staff passwords (melzayat, msoubhy) in every system they touch	Mohamed	Written confirmation
1	Remove hardcoded passwords from `server/storage.ts`; deploy	Tech lead	Grep confirms removal
2	Rotate: SESSION_SECRET, VAPID keys, all Paymob credentials, Upstash tokens, Discord bot token, R2 keys	Tech lead	Rotation log
2	Move VAPID_PRIVATE_KEY to encrypted Secrets	Tech lead	Screenshot confirmation
3	Add `.env`, `*.sqlite`, `*.db`, `logs/`, `uploads/`, `secrets/` to `.gitignore`	Tech lead	Commit + diff
3	Run `gitleaks` / `trufflehog` full history scan	Tech lead	Report
4	Write startup secret-presence validator that fails boot if any critical env var is missing	Tech lead	Boot log shows check
5	Fix rate limiter path: `/api/admin/login` → `/api/admin-auth/login`	Tech lead	Test 21 rapid logins blocked
5	Fix `/api/admin/cylinders/:id/sensor` middleware	Tech lead	Consumer test returns 403

Week 2 — Money Integrity + Observability

Day	Action	Owner	Verification

6	Add `paymob_transaction_id UNIQUE` column; migration	Tech lead	Migration applied
7	Rewrite Paymob webhook handler: idempotency check + unique constraint upsert	Tech lead	Unit test: same webhook twice = one ledger entry
8	Remove `price` and `paymentStatus` from `createOrderSchema.pick()`	Tech lead	Schema diff
8	Remove hardcoded Paymob iframe ID fallback	Tech lead	Boot fails without env var
9	Build daily Paymob reconciliation job against `/api/acceptance/transactions`	Tech lead	First run report
10	Install `unhandledRejection` + `uncaughtException` handlers, log to Sentry	Tech lead	Test rejection captured
11	Fix `autoDispatchOrder` silent catch — log + Sentry capture	Tech lead	Forced failure visible
12	Convert inline `res.status(500)` to `next(err)` across top 20 highest-traffic routes	Tech lead	Sentry event count increase
12	Install UptimeRobot (or Better Stack) pinging `/api/health` every 60s	Tech lead	Test alert received

Week 3 — Data Protection + Dependencies

Day	Action	Owner	Verification
13	Build daily `pg_dump` to R2 with 30-day retention	Tech lead	First backup in R2
14	Write and test restore procedure to a fresh Neon branch	Tech lead	Written runbook
15	Upgrade drizzle-orm ≥0.45.2	Tech lead	Regression suite green
15	Force axios override ≥1.15.0 via `overrides` in package.json	Tech lead	`npm audit` shows fix
16	Update multer, vite, force-resolve lodash/path-to-regexp/undici/rollup	Tech lead	`npm audit` clean on high

17	Remove `@anthropic-ai/claude-code` and `@expo/ngrok` from production deps	Tech lead	Bundle size drop
18	Application-layer encryption for `drivers.nationalId`	Tech lead	Migration + backfill
19	Hash or purge `orders.otpCode` post-delivery	Tech lead	Test: delivered order has null/hashed OTP

Week 4 — Discipline + Migration Decision

Day	Action	Owner	Verification
20	Set up GitHub Actions: lint + typecheck + regression tests on every PR	Tech lead	First PR blocked by failing check
21	Create `main` branch protection: require PR, require passing checks, no direct push	Tech lead	Direct push rejected
22	Create `develop` branch; enforce PR-based merge to `main`	Tech lead	Written workflow doc
23	One-page incident runbook in Notion	Tech lead	Document linked in board log
24	Load test: simulate 100, 500, 1000 concurrent users against staging	Tech lead	Breaking point identified
25	Migration target decision: Cloudflare Workers + D1/R2 vs Railway/Render vs self-managed VPS	Mohamed + Tech lead	Written decision in board log
26	Admin login logout `session.destroy()` fix	Tech lead	Session invalidated test
27	In-memory OTP → Redis migration	Tech lead	OTP survives restart
28	Final audit re-run against this scorecard	Mohamed	Updated scorecard
30	Go/no-go decision for Cairo Phase 1 pilot	Mohamed	Board log entry

---

Migration Readiness Verdict

The codebase is worth migrating as-is. The domain model is real, the data layer is sound, the state machines are correctly architected, the ledger design is appropriate for a regulated LPG business. This is not a Replit prototype dressed up — it is a serious platform that happens to run on Replit.

What should migrate unchanged:

- Database schema and migrations (Neon Postgres is already external; it comes with you)
- Drizzle ORM layer (after the security upgrade)
- Business logic: cylinder lifecycle, order state machine, custody events, ledger tables
- R2 object storage and Upstash Redis (already external services)
- The three-flow auth architecture (consumer OIDC / mobile bearer / admin session)

What should be refactored during migration:

- Logging: introduce Pino with structured JSON + request IDs
- Error handling: convert inline `res.status(500)` to `next(err)` uniformly
- Background workers: move SLA engine, Slack bot, Discord bot out of the web process onto a dedicated worker deployment (they are currently coupled to the web process, which is incompatible with Autoscale's horizontal scaling model — every replica will run its own SLA loop and duplicate work)
- Distributor tier hierarchy: implement in schema as a first-class field
- Penalty matrix: convert to rule-engine in code

Critical note on Autoscale and background workers: Your current `.replit` uses `deploymentTarget = "autoscale"` but your background workers (`startSLAEngine`, `startDiscordBot`, `startSlackBot`, in-process `setInterval`s) live inside the web server process. On Autoscale, every replica will independently run these loops, producing duplicate SLA escalations, duplicate Slack messages, and race conditions in the reconciliation logic. **This is a latent defect that only appears under the load conditions of a national launch.** It is the single strongest argument for moving to a deployment model where web and worker processes are separate — and it is not something you can paper over on Replit.

Recommended migration target: Railway or Render for the web + worker split with managed Postgres continuity (Neon stays), or Cloudflare Workers + Queues + D1 if you want a more radical re-architecture aligned with the Cloudflare R2 you already use. Given timeline pressure for the 6-month launch, **Railway is the lower-risk choice**: it supports the current Node/Express/Drizzle stack without modification, separates web and worker processes cleanly, and gives you real CI/CD.

Next Actions (This Week)

1. **Revoke GitHub PAT. Today. Before you close this document.** Then re-add the remote cleanly.
2. **Change the two staff passwords** in every system they might have been reused on.
3. **Inspect commit `f1db226`** and decide on Cloudflare token rotation.
4. **Scope the Paymob idempotency fix** — it is one column, one migration, one handler rewrite. Small change, enormous risk reduction.
5. **Decide this week** whether to commit to Railway migration on a 30–45 day timeline. The Autoscale worker-duplication defect means the current deployment model has a ceiling you will hit in Phase 1 regardless of everything else in this document.

One hard truth before you close this: the audit results reveal that you have been building a genuinely serious platform — and then leaving the front door open. The code is disciplined. The credentials, secrets, and git hygiene are not. The gap between those two is the gap between a company that can launch nationally and a company that will make headlines for the wrong reasons. Every Tier 0 item in this report is a decision, not a development task. Decide this week.