2026-07-16

RAPID PLATFORM AUDIT

NEXT.JS SAAS STARTER (SAMPLE)

13-LAYER AUDIT

AI DIRECTED ENGINEERING STACK ASSESSMENT

AUDIT ID	RPA-98569
PRODUCT	Next.js SaaS Starter (Sample)
DEVELOPER	Next.js SaaS Starter (Sample)
REPO	github.com/nextjs/saas-starter (PUBLIC)
LIVE	next-saas-start.vercel.app
DATE	2026-07-16
AUDIT TYPE	Full Codebase — repo access + live deployment

RED — ACTIVE VULNERABILITIES, NEEDS OPS HARDENING

2	9	2
PASS	WARNING	CRITICAL

01 PRODUCT SUMMARY

SOLID HAND-ROLLED AUTH AND STRIPE WEBHOOK INTEGRITY; CRITICAL GAPS ARE DEPENDENCY CVES, MISSING CI/CD GOVERNANCE, NO RATE LIMITING, AND ABSENT OPERATIONAL MONITORING.

Next.js SaaS Starter (Sample) is a software platform built by Next.js SaaS Starter (Sample). Code quality is ahead of infrastructure maturity. The app has no secrets exposed, good server-side auth, and correct webhook signature verification — but is running a canary Next.js with active HIGH CVEs, has no CI pipeline, no rate limiting on any endpoint, and zero production monitoring. Needs ops hardening, not a rewrite.

Next.js 15 App Router with Tailwind and Radix UI renders correctly on the live probe with zero console errors. No error boundaries are visible on route segments — a white-screen risk if a server component throws. No evidence of `next/image` usage in the scanned file list; images appear unoptimised. No lazy-loading or route-based code-splitting beyond what Next.js provides by default. All form actions use Zod validation via the `validatedAction` / `validatedActionWithUser` wrappers (lib/auth/middleware.ts). The Stripe webhook correctly verifies signatures using `stripe.webhooks.constructEvent` before processing events. Error shapes return structured JSON. Business logic is separated into service-layer queries (lib/db/queries.ts). No raw stack traces observed in response handlers.

This audit assessed production readiness across Faction's 13-layer stack for the solo tier.

02 WHAT WE REVIEWED

We audited the full GitHub repository with deep code-level access. The repo (https://github.com/nextjs/saas-starter) was reviewed for architecture decisions, code quality, dependency hygiene, security posture, database design, and production readiness. Every finding is mapped to the Faction 13-Layer AI Directed Engineering Technology Stack. We also tested the live deployment at https://next-saas-start.vercel.app/.

METHODOLOGY

We examined repository structure, CI/CD configuration, authentication and authorization patterns, data layer migrations and policies, API security, hosting and deployment controls, observability, rate limiting, caching, scaling posture, and disaster recovery readiness. Areas we could not fully assess: Branch protection enforcement — GitHub API not accessible; cannot confirm main branch requires CI checks to pass; Database backup configuration — operator-hosted PostgreSQL; no backup policy visible in repo; Staging environment existence — no vercel.json or staging URL documented in repo; may exist as operator config outside the codebase; Production env var configuration — cannot verify POSTGRES_URL, AUTH_SECRET strength, or STRIPE_WEBHOOK_SECRET are correctly set in the Vercel project settings

03 EXECUTIVE SUMMARY

SOLID HAND-ROLLED AUTH AND STRIPE WEBHOOK INTEGRITY; CRITICAL GAPS ARE DEPENDENCY CVES, MISSING CI/CD GOVERNANCE, NO RATE LIMITING, AND ABSENT OPERATIONAL MONITORING.

The application has clean server-side auth, correct webhook signature verification, zero committed secrets, parameterized queries, and a well-structured service layer.

The critical gaps are entirely operational: a canary Next.js with active HIGH CVEs deploying with no CI gate, no rate limiting on any endpoint including login, no error tracking, no uptime monitoring, and no staging environment.

These are activation and configuration items, not architectural rewrites. Resolve the P0 (add CI and block the CVE-laden canary) and the two P1s (rate limiting on auth, upgrade drizzle-orm) before accepting production traffic at any volume.

FINDING SEVERITY SUMMARY

SEVERITY	COUNT	DESCRIPTION
CRITICAL	0	
HIGH	5	drizzle-orm v0.43.1 has HIGH SQL injection CVE (GHSA-gpj5-g38j-94v9), Hand-rolled auth with no managed provider — password reset and account takeover paths unverified, No CI pipeline — HIGH CVE dependencies ship to production with zero automated gate, next@15.6.0-canary.59 has multiple HIGH/MODERATE CVEs — should not be used in production, No rate limiting on sign-in / sign-up — brute-force and credential-stuffing exposure
MEDIUM	7	No React error boundaries on dashboard route segment, No visible backup configuration for production database, No staging environment or documented deployment strategy, Missing X-Frame-Options and CSP security headers, No rate limiting on Stripe checkout or team/user API endpoints, No error tracking configured — production exceptions are invisible, No health endpoint and no external uptime monitoring
LOW	1	No Cache-Control headers on API routes; no next/image for image optimisation

04 13-LAYER SCORECARD

01	FRONTEND FOUNDATIONS	Next.js 15 App Router with Tailwind and Radix UI renders correctly on the live probe with zero console errors. No error boundaries are visible on route segments — a white-screen risk if a server component throws. No evidence of `next/image` usage in the scanned file list; images appear unoptimised. No lazy-loading or route-based code-splitting beyond what Next.js provides by default.	WARNING
02	APIS & BACKEND LOGIC	All form actions use Zod validation via the `validatedAction` / `validatedActionWithUser` wrappers (lib/auth/middleware.ts). The Stripe webhook correctly verifies signatures using `stripe.webhooks.constructEvent` before processing events. Error shapes return structured JSON. Business logic is separated into service-layer queries (lib/db/queries.ts). No raw stack traces observed in response handlers.	PASS
03	DATABASE & STORAGE	PostgreSQL via Drizzle ORM with parameterized queries — no SQL injection risk in application code. Migrations exist as SQL files under lib/db/migrations/. However the schema uses no Row-Level Security (this is a direct Postgres deployment, not Supabase), so application-layer tenant isolation is the only guard. The drizzle-orm dependency (v0.43.1) has a HIGH severity CVE (GHSA-gpj5-g38j-94v9) for SQL injection via improperly escaped identifiers when untrusted input reaches `sql.identifier()` or `.as()`. No backup configuration is visible in the repo.	WARNING
04	AUTH & PERMISSIONS	Auth is hand-rolled using bcryptjs for password hashing and jose JWTs stored in httpOnly secure cookies — functionally sound but not a managed provider. Server-side session verification occurs in middleware.ts and queries.ts before any data access. Role field exists on users and teamMembers tables. No audit log for sensitive actions beyond the activityLogs table (which does log key events). No re-authentication gate before destructive actions like account deletion. The `AUTH_SECRET` key is read from env at module load — if it is short or absent in production the JWT signature provides no security.	WARNING
05	HOSTING & DEPLOYMENT	App is deployed on Vercel (live probe confirms). Custom domain resolves with HSTS present. No staging environment, preview deployment configuration, or rollback plan is documented in the repo. No vercel.json or equivalent deployment config found in scanned files. Environment variable separation is operator-dependent with no documented per-environment strategy. No CI pipeline means any commit can reach production ungated.	WARNING

06	**CLOUD & COMPUTE**	Serverless-first on Vercel (Next.js). No AI/LLM inference in this codebase — the app is a SaaS starter with Stripe billing, so L06 AI cost caps are not applicable. No budget alerts or cost monitoring configuration is visible in the repo. Platform limits are undocumented.	WARNING
07	**CI/CD & VERSION CONTROL**	No CI pipeline exists — the .github/workflows/ directory is entirely absent from the scanned file list. There is no lint, type-check, test, or build step that runs automatically before deployment. Branch protection cannot be confirmed as enforced. The OSV scanner found a HIGH CVE in the core ORM dependency (drizzle-orm 0.43.1) and multiple HIGH/MODERATE CVEs in next@15.6.0-canary.59, none of which are blocked by any automated gate. No Dependabot or Renovate configuration is present.	CRITICAL
08	**SECURITY & RLS**	Gitleaks found zero committed secrets — .env is gitignored, env vars used correctly. Semgrep found no injection or XSS vectors in application code. HTTPS is enforced (HSTS present). However: no CSP or X-Frame-Options headers (live probe confirms), the hand-rolled auth means the TypeScript model is the only isolation layer with no SQL RLS as a defence-in-depth backstop, and the HIGH CVE in drizzle-orm's identifier escaping is a latent injection risk. At solo-tier, missing CSP is deferred to growth readiness, but X-Frame-Options absence allows clickjacking.	WARNING
09	**RATE LIMITING**	No rate limiting of any kind is present in the codebase. The sign-in and sign-up server actions, the Stripe checkout endpoint, and all team/user API routes have zero throttling. This exposes the auth endpoints to brute-force and credential stuffing attacks. No middleware, no in-memory limiter, no edge-level limiting.	CRITICAL
10	**CACHING & CDN**	Vercel's CDN handles static asset delivery as a platform default. No explicit `Cache-Control` headers are set on API routes or edge responses. No `next/image` usage is visible in scanned files — images appear to be served unoptimised. No ISR or SWR cache strategy is documented. No user-data cache isolation risk detected (no shared cache of authenticated content observed).	WARNING
11	**LOAD BALANCING & SCALING**	Serverless on Vercel — horizontal auto-scaling is platform-managed. The `postgres` npm package (v3.4.5) handles connection pooling for the PostgreSQL connection (it uses a connection pool by default). No connection exhaustion risk at solo scale. No load testing evidence, but at solo tier with serverless compute this is a next-tier concern.	PASS

12	ERROR TRACKING & LOGS	No error tracking SDK is installed — package.json contains no error tracking dependency. Logging is via `console.error` in the webhook handler and middleware (e.g., middleware.ts:35, webhook/route.ts:16). No structured logging, no log levels, no alerting on critical errors. No React error boundaries on major routes. There is no production error visibility beyond what the hosting platform may surface in its logs.	WARNING
13	AVAILABILITY & RECOVERY	The live probe confirms the production URL resolves and returns 200. However, there is no external uptime monitoring configured (no probe URL or health endpoint in the repo), no health check endpoint, no incident response documentation, no DR playbook, and no evidence of a backup restore test. The app depends entirely on Vercel and the operator's database provider for availability — no SLA or recovery plan is documented.	WARNING

Production-Ready Layers: 2 | Needs Work: 9 | Critical/Weak: 2

05 DETAILED FINDINGS

01 FRONTEND FOUNDATIONS

WARNING

Next.js 15 App Router with Tailwind and Radix UI renders correctly on the live probe with zero console errors. No error boundaries are visible on route segments — a white-screen risk if a server component throws. No evidence of `next/image` usage in the scanned file list; images appear unoptimised. No lazy-loading or route-based code-splitting beyond what Next.js provides by default.

FINDING

NO REACT ERROR BOUNDARIES ON DASHBOARD ROUTE SEGMENT

No error.tsx exists under app/(dashboard)/ so server-component errors produce a white screen. Users lose the entire dashboard on any transient server error; React error boundaries allow graceful fallback UI.

HOW TO FIX

Create app/(dashboard)/error.tsx exporting a default React client component that accepts `error` and `reset` props and renders a user-friendly fallback with a retry button.

03 DATABASE & STORAGE

WARNING

PostgreSQL via Drizzle ORM with parameterized queries — no SQL injection risk in application code. Migrations exist as SQL files under lib/db/migrations/. However the schema uses no Row-Level Security (this is a direct Postgres deployment, not Supabase), so application-layer tenant isolation is the only guard. The drizzle-orm dependency (v0.43.1) has a HIGH severity CVE (GHSA-gpj5-g38j-94v9) for SQL injection via improperly escaped identifiers when untrusted input reaches `sql.identifier()` or `.as()`. No backup configuration is visible in the repo.

FINDING

DRIZZLE-ORM V0.43.1 HAS HIGH SQL INJECTION CVE (GHSA-GPJ5-G38J-94V9)

drizzle-orm@0.43.1 in pnpm-lock.yaml is vulnerable to SQL injection via improperly escaped identifiers (GHSA-gpj5-g38j-94v9, CVSS 7.5). If any query dynamically builds identifiers or aliases from request parameters, attackers can exfiltrate or manipulate data. CVSS 7.5 / HIGH.

HOW TO FIX

In package.json change `"drizzle-orm": "^0.43.1"` to `"drizzle-orm": "^0.45.2"`, then run `pnpm install` and regression-test all DB queries.

FINDING

NO VISIBLE BACKUP CONFIGURATION FOR PRODUCTION DATABASE

No backup policy is configured or documented for the production PostgreSQL instance. Without confirmed backups, a database corruption or accidental deletion event results in permanent data loss.

HOW TO FIX

Enable automated daily snapshots on the hosting provider's database dashboard; document the backup schedule, retention period, and restore procedure in a RUNBOOK.md file in the repo root.

---

04 AUTH & PERMISSIONS

WARNING

Auth is hand-rolled using bcryptjs for password hashing and jose JWTs stored in httpOnly secure cookies — functionally sound but not a managed provider. Server-side session verification occurs in middleware.ts and queries.ts before any data access. Role field exists on users and teamMembers tables. No audit log for sensitive actions beyond the activityLogs table (which does log key events). No re-authentication gate before destructive actions like account deletion. The `AUTH_SECRET` key is read from env at module load — if it is short or absent in production the JWT signature provides no security.

FINDING

HAND-ROLLED AUTH WITH NO MANAGED PROVIDER — PASSWORD RESET AND ACCOUNT TAKEOVER PATHS UNVERIFIED

No password reset endpoint, no email verification, and no account lockout exist in the custom auth implementation. Custom auth implementations frequently miss edge cases (timing attacks, account enumeration, insecure reset flows). A missing password-reset flow means users are permanently locked out if they forget credentials, and no lockout means the login endpoint is vulnerable to brute-force.

HOW TO FIX

(a) Add a forgot-password page + server action that emails a signed, time-limited reset token (valid ≤ 1 hour) and a reset-password page that verifies it; (b) add server-side rate limiting on the sign-in action (see L9 finding); (c) alternatively, replace custom auth with a managed provider integration.

---

05 HOSTING & DEPLOYMENT

WARNING

App is deployed on Vercel (live probe confirms). Custom domain resolves with HSTS present. No staging environment, preview deployment configuration, or rollback plan is documented in the repo. No vercel.json or equivalent deployment config found in scanned files. Environment variable separation is operator-dependent with no documented per-environment strategy. No CI pipeline means any commit can reach production ungated.

FINDING

NO STAGING ENVIRONMENT OR DOCUMENTED DEPLOYMENT STRATEGY

No staging Vercel project or PR preview environment exists; production receives every main-branch push ungated. Schema migrations and breaking API changes hit production users immediately with no test gate. A bad deploy cannot be quickly rolled back without a documented plan.

HOW TO FIX

Create a second Vercel project connected to the same repo targeting a staging branch; configure `pnpm db:migrate` to run against a separate staging database via different env vars; add a RUNBOOK.md documenting the deploy checklist and rollback steps (Vercel instant rollback command).

---

07 CI/CD & VERSION CONTROL

CRITICAL

No CI pipeline exists — the .github/workflows/ directory is entirely absent from the scanned file list. There is no lint, type-check, test, or build step that runs automatically before deployment. Branch protection cannot be confirmed as enforced. The OSV scanner found a HIGH CVE in the core ORM dependency (drizzle-orm 0.43.1) and multiple HIGH/MODERATE CVEs in next@15.6.0-canary.59, none of which are blocked by any automated gate. No Dependabot or Renovate configuration is present.

FINDING

NO CI PIPELINE — HIGH CVE DEPENDENCIES SHIP TO PRODUCTION WITH ZERO AUTOMATED GATE

No .github/workflows/ CI file exists; HIGH CVE dependencies ship ungated to production. Without CI, broken code, unpatched CVEs, and type errors deploy directly to users. The rubric rates absence of CI with known critical/high CVEs as P0.

HOW TO FIX

Create .github/workflows/ci.yml that runs on pull_request and push to main with jobs: (1) `pnpm install --frozen-lockfile`, (2) `pnpm tsc --noEmit`, (3) `pnpm build`, (4) `pnpm osv-scanner --lockfile pnpm-lock.yaml --fail-on-vuln`. Enable branch protection on main requiring all CI checks to pass before merge.

FINDING

NEXT@15.6.0-CANARY.59 HAS MULTIPLE HIGH/MODERATE CVES — SHOULD NOT BE USED IN PRODUCTION

next@15.6.0-canary.59 is a pre-release build with GHSA-5j59-xgg2-r9c4 and GHSA-h25m-26qc-wcjf (both HIGH CVSS 7.5) enabling unauthenticated DoS on Server Action endpoints. Canary builds are not production-ready; the HIGH CVEs enable unauthenticated denial-of-service by sending crafted requests to Server Action endpoints. Any user can crash the server process.

HOW TO FIX

In package.json change `"next": "15.6.0-canary.59"` to the latest stable Next.js 15 release (≥ 15.6.0-canary.61 at minimum, or latest stable 15.x), then run `pnpm install` and full regression test.

---

08 SECURITY & RLS

WARNING

Gitleaks found zero committed secrets — .env is gitignored, env vars used correctly. Semgrep found no injection or XSS vectors in application code. HTTPS is enforced (HSTS present). However: no CSP or X-Frame-Options headers (live probe confirms), the hand-rolled auth means the TypeScript model is the only isolation layer with no SQL RLS as a defence-in-depth backstop, and the HIGH CVE in drizzle-orm's identifier escaping is a latent injection risk. At solo-tier, missing CSP is deferred to growth readiness, but X-Frame-Options absence allows clickjacking.

FINDING

MISSING X-FRAME-OPTIONS AND CSP SECURITY HEADERS

Production responses have no X-Frame-Options or CSP header; live_probe confirms both absent. Clickjacking can trick logged-in users into performing unintended actions (e.g., cancelling subscription, inviting attackers as team members). CSP is a growth-tier expectation but X-Frame-Options is a trivially added one-liner.

HOW TO FIX

In next.config.ts add a `headers()` export with at minimum: `X-Frame-Options: SAMEORIGIN`, `X-Content-Type-Options: nosniff`, `Referrer-Policy: strict-origin-when-cross-origin`. Add a basic CSP in report-only mode initially then tighten to enforcing.

---

09 RATE LIMITING

CRITICAL

No rate limiting of any kind is present in the codebase. The sign-in and sign-up server actions, the Stripe checkout endpoint, and all team/user API routes have zero throttling. This exposes the auth endpoints to brute-force and credential stuffing attacks. No middleware, no in-memory limiter, no edge-level limiting.

FINDING

NO RATE LIMITING ON SIGN-IN / SIGN-UP — BRUTE-FORCE AND CREDENTIAL-STUFFING EXPOSURE

app/(login)/actions.ts signIn and signUp actions have no rate limiting; unlimited password attempts are possible. Unthrottled login endpoints are the primary credential-stuffing attack surface. With bcrypt hashing (10 rounds), each attempt also consumes CPU proportional to load.

HOW TO FIX

Add a rate-limiting check at the top of the signIn server action. For serverless Next.js, use an external rate-limit store (e.g., a Redis-compatible KV from the hosting provider) keyed on `ip + 'login'` with a limit of e.g. 5 attempts per 15 minutes; return a structured error with a `429` equivalent when exceeded. Apply a looser limit to signUp.

FINDING

NO RATE LIMITING ON STRIPE CHECKOUT OR TEAM/USER API ENDPOINTS

app/api/stripe/checkout/route.ts and app/api/team/route.ts have no per-user rate limits. Unthrottled Stripe checkout creation can generate large numbers of incomplete payment attempts, creating noise and potential Stripe rate-limit violations. Team mutation APIs without limits allow data abuse.

HOW TO FIX

Create a rate-limit middleware helper that reads the session user ID and applies a token-bucket or sliding-window limit (e.g., 10 checkout attempts per hour per user) backed by an external KV store. Apply this middleware at the top of each POST handler.

---

10 CACHING & CDN

WARNING

Vercel's CDN handles static asset delivery as a platform default. No explicit `Cache-Control` headers are set on API routes or edge responses. No `next/image` usage is visible in scanned files — images appear to be served unoptimised. No ISR or SWR cache strategy is documented. No user-data cache isolation risk detected (no shared cache of authenticated content observed).

FINDING

NO CACHE-CONTROL HEADERS ON API ROUTES; NO NEXT/IMAGE FOR IMAGE OPTIMISATION

No Cache-Control header on authenticated API routes and no next/image usage. Without `no-store` on authenticated API responses, edge caching can inadvertently serve one user's data to another. Missing image optimisation increases page weight unnecessarily.

HOW TO FIX

Add `headers: { 'Cache-Control': 'no-store' }` to all NextResponse.json() calls in app/api/* that return user-specific data. Replace any <img> tags in components with next/image.

---

12 ERROR TRACKING & LOGS

WARNING

No error tracking SDK is installed — package.json contains no error tracking dependency. Logging is via `console.error` in the webhook handler and middleware (e.g., middleware.ts:35, webhook/route.ts:16). No structured logging, no log levels, no alerting on critical errors. No React error boundaries on major routes. There is no production error visibility beyond what the hosting platform may surface in its logs.

FINDING

NO ERROR TRACKING CONFIGURED — PRODUCTION EXCEPTIONS ARE INVISIBLE

No error tracking SDK is installed; production exceptions are invisible. Without error tracking, production bugs go undetected until users report them. Silent failures in payment flows or auth can cause revenue loss and churn before the team is aware.

HOW TO FIX

Add an error tracking SDK to package.json, initialise it in app/layout.tsx (client) and in a server instrumentation file (instrumentation.ts for Next.js), configure the DSN via an environment variable (e.g., ERROR_TRACKING_DSN), and add a global error.tsx boundary that reports uncaught errors.

---

13 AVAILABILITY & RECOVERY

WARNING

The live probe confirms the production URL resolves and returns 200. However, there is no external uptime monitoring configured (no probe URL or health endpoint in the repo), no health check endpoint, no incident response documentation, no DR playbook, and no evidence of a backup restore test. The app depends entirely on Vercel and the operator's database provider for availability — no SLA or recovery plan is documented.

FINDING

NO HEALTH ENDPOINT AND NO EXTERNAL UPTIME MONITORING

No /api/health route exists and no external uptime monitor is configured, so production outages are invisible until user reports. Outages go undetected until users complain. A health endpoint enables monitoring tools and load balancers to detect availability issues automatically.

HOW TO FIX

Create app/api/health/route.ts that runs a lightweight DB query (e.g., `SELECT 1`) and returns `{status:'ok'}` with HTTP 200, or `{status:'error'}` with HTTP 503 on failure. Then configure an external uptime monitoring service pointed at `https://<domain>/api/health` with a check interval of ≤ 5 minutes and alert routing to the team's primary contact.

06 PRIORITIZED FINDINGS

13 ITEMS, RANKED BY IMPACT

PRIORITY	FINDING	LAYER
P1	drizzle-orm v0.43.1 has HIGH SQL injection CVE (GHSA-gpj5-g38j-94v9)	Database & Storage
P1	Hand-rolled auth with no managed provider — password reset and account takeover paths unverified	Auth & Permissions
P1	No CI pipeline — HIGH CVE dependencies ship to production with zero automated gate	CI/CD & Version Control
P1	next@15.6.0-canary.59 has multiple HIGH/MODERATE CVEs — should not be used in production	CI/CD & Version Control
P1	No rate limiting on sign-in / sign-up — brute-force and credential-stuffing exposure	Rate Limiting
P2	No React error boundaries on dashboard route segment	Frontend Foundations
P2	No visible backup configuration for production database	Database & Storage
P2	No staging environment or documented deployment strategy	Hosting & Deployment
P2	Missing X-Frame-Options and CSP security headers	Security & RLS
P2	No rate limiting on Stripe checkout or team/user API endpoints	Rate Limiting
P2	No error tracking configured — production exceptions are invisible	Error Tracking & Logs
P2	No health endpoint and no external uptime monitoring	Availability & Recovery
P3	No Cache-Control headers on API routes; no next/image for image optimisation	Caching & CDN

07 HOW WE WORK

• 13-Layer Stack Coverage — Every engagement evaluates the full stack, from frontend component architecture to disaster recovery.

• AI-Native Engineering — We build with AI agents as first-class infrastructure components, not bolted-on features.

• Implementation-Ready Recommendations — Every finding includes a specific, actionable fix — not generic advice.

08 NEXT STEP

Book a follow-up call to walk through your findings:

WE BUILD THE THINGS THAT ACTUALLY WORK.

gofactiongroup.com