SEO Readiness Portal — Real Audit Engine (2026-06-02)

SEO Readiness Portal — Real Audit Engine (2026-06-02)

Status: DEPLOYED to production **Scope:** MC #102800 / #102801 / #102802 / #102803 — Real live crawl audit runner (replaces local readiness stub) **Deploy date:** 2026-06-02 **Evidence:** `/tmp/alai/996bd450/evidence-102800/verification.json`, `/tmp/alai/996bd450/evidence-102820/verification.json` **Image:** `alairegistry.azurecr.io/seo-readiness-portal:20260602-real-audit`

Overview

The SEO Readiness Portal now performs **real live HTTP crawl audits** against client websites, replacing the previous local form-validation-only stub. The audit engine fetches the home page, robots.txt, and sitemap.xml from the public internet, parses them with **cheerio** (HTML5-aware DOM parser), and emits **P0/P1/P2 findings** based on industry-standard SEO readiness signals.

All findings flow into the backlog system (Phase 4) and feed the client report generator (Phase 5). Reports are exported as Markdown and include a mandatory no-ranking-guarantee disclaimer.

What changed: Phase 3 (audit runner), Phase 4 (findings/backlog), and Phase 5 (report generation) are now **REAL** — they operate on live crawl data, not local form fields. The previous Phase 4–11 local readiness workflow is retained as a fallback mode (`mode: "local_readiness"` vs `mode: "live_crawl"`).

Architecture

```mermaid
flowchart LR
    A[Operator Browser] -->|HTTPS + CF Access| B[Cloudflare Access]
```

```
    B -->|Authenticated header| C[Azure App Service
seo-readiness-alai]
    C -->|Next.js Server Action| D[Live Crawl Runner]
    D -->|SSRF-guarded fetch| E[Client website]
    D -->|cheerio parse| F[Findings + Backlog]
    F --> G[Report Generator]
    G --> H[Markdown Export]
    C -->|Write| I[/home/data/workspace.json]
    C -->|Write| J[/home/data/audits/auditId.json]
`
```

Components

| Component | Technology | Purpose | Location | |-----------|-----------|---------|----------| | **Live Crawl Runner** | TypeScript + Node.js fetch | Fetch home/robots/sitemap, parse with cheerio, emit findings | src/lib/audit/runner.ts | | **SSRF Guard** | Custom URL validation + AbortController | Block private IPs, enforce 9s per-fetch + 45s total timeout, 2 MB body cap | src/lib/audit/crawl-guard.ts | | **HTML Parser** | cheerio (HTML5 mode) | Parse title, meta, headings, links, canonical, OG tags | src/lib/audit/crawl-parser.ts | | **Findings Engine** | TypeScript | Emit P0/P1/P2 findings with evidence JSON, block forbidden ranking claims | src/lib/audit/runner.ts (liveFinding) | | **Backlog Generator** | TypeScript | Convert findings → backlog items, enforce evidence-URL for done gate | src/lib/reports/generator.ts | | **Report Generator** | TypeScript | Generate client-facing Markdown report with no-ranking disclaimer | src/lib/reports/generator.ts | | **Persistence** | JSON file backend | Atomic write to /home/data/workspace.json + /home/data/audits/.json | src/lib/workspace/persistence.ts |

Data Flow

1. **Operator triggers audit** (authenticated browser at https://seo-tools.alai.no/partners)
2. **Server Action calls runLiveCrawlAudit()** with client, site, now
3. **guardedFetch()** retrieves home page, robots.txt, sitemap.xml with SSRF guard + timeout
4. **cheerio** parses HTML5-compliant DOM (handles broken HTML gracefully)
5. **Findings emitted** — P0/P1/P2 severity, 11 categories (crawlability, indexability, content, technical, metadata, performance, mobile, accessibility, structure, security, evidence)
6. **Atomic write** — audit JSON → /home/data/audits/.json, workspace update → /home/data/workspace.json
7. **Backlog items generated** from findings (operator can convert any finding to a backlog task)
8. **Report generated** from audit + backlog, no-ranking disclaimer injected
9. **Markdown export** with checksum and handoff checklist

SSRF Guard

The crawl engine protects against **Server-Side Request Forgery (SSRF)** attacks:

Blocked targets

- Non-http(s) schemes (e.g., file:// , ftp:// , gopher://)
- Bare IP literals (http://192.168.1.1/ , http://[::1]/)
- Private IPv4 ranges: 10.0.0.0/8 , 172.16.0.0/12 , 192.168.0.0/16 , 127.0.0.0/8 , 169.254.0.0/16 (includes cloud metadata endpoint 169.254.169.254)
- Private IPv6 ranges: ::1 , fc00::/7 , fe80::/10
- Numeric/encoded IP hostnames (e.g., 0x7f.0.0.1 , 2130706433)

Timeouts

- **Per-fetch:** 9 seconds (home, robots, sitemap fetched sequentially)
- **Total audit:** 45 seconds hard limit (AbortController abort on timeout)
- **Body size cap:** 2 MB (drains and cancels response body on overflow to prevent socket leaks)

Known limitations (CEO decision: acceptable for MVP)

- DNS rebind protection deferred — the guard covers literal IPs but does not resolve hostnames at validation time (a follow-on MC can add dns.lookup pre-check)
- No per-operator rate limiting (deferred to follow-on MC)
- Single-writer assumption: if two Azure App Service instances concurrently trigger crawls, last write wins on workspace.json (Postgres migration is a follow-on MC)

File-backed Persistence

The audit engine writes to **persistent App Service storage** (Azure flag WEBSITES_ENABLE_APP_SERVICE_STORAGE=true):

- **Workspace state:** /home/data/workspace.json (atomic write with temp + rename, 8 KB typical size)
- **Audit archives:** /home/data/audits/.json (one file per audit, ~20–50 KB per file)

Why file-backend for MVP: CEO decision (a) — Postgres migration is a follow-on MC. File backend is deterministic, testable, and works for single-operator phase. Concurrent writes from multiple Azure instances are NOT handled (last write wins). **Atomic write protocol:** 1. Write to temp file: /home/data/workspace.json.tmp-
2. fs.rename() to /home/data/workspace.json (atomic on POSIX)

3. Collision-safe audit IDs: audit----<6charUUID>

Findings Categories and Severity

The live crawl audit emits **P0 (blocker), P1 (high), P2 (medium)** findings across **11 categories**:

| Category | P0 Findings | P1 Findings | P2 Findings | |----------|-------------|-------------|-------------| | **crawlability** | robots.txt blocks all crawlers, home page 403/503/429 | robots.txt fetch failed | Crawl-delay > 60s | | **indexability** | Home status ≠ 200, robots meta noindex | | | | **content** | Missing h1, title missing | Title < 30 or > 70 chars, h1 ≠ title | Meta description < 120 or > 160 chars, missing priority services | | **technical** | | Missing viewport, sitemap index (nested, not flat) | og:image is relative URL | | **metadata** | | Missing meta description, canonical mismatch | Missing og:title, og:description, or og:image | | **performance** | | | href=# placeholder links (> 5) | | **mobile** | | | No viewport | | **accessibility** | | | Images missing alt (> 5) | | **structure** | | | External links < 3 (isolation signal) | | **security** | Canonical URL is http:// (not https://) | | | | **evidence** | | | Analytics/Search Console status unknown |

Forbidden claim words: The generator enforces a hard block on ranking, rankings, traffic lift, traffic growth, guarantee, guaranteed in all finding/backlog/report text. Any match throws an error and aborts the audit.

Findings ? Backlog ? Report Flow

1. **Audit emits findings** — JSON array with

{ id, severity, category, title, description, recommendation, evidence }
2. **Operator converts finding to backlog item** (optional — not all findings require action)
3. **Backlog item fields:**
   - title: "Resolve {severity} {category} readiness item: {finding.title}"
   - notes: "{finding.recommendation} This is a readiness task from local workspace evidence only."
   - status: "open" | "in_progress" | "done" | "wont_fix"
   - evidenceUrl: REQUIRED for status: "done" (external proof the issue was fixed)
4. **Report generator** pulls latest audit + backlog, emits Markdown with:
   - Audit metadata (date, mode, status, findings count)
   - Scope section: "This report reflects basic public-page observability. It does not use Google Search Console, Analytics, paid keyword APIs, or private CMS data. Findings are readiness signals only. **This assessment does not predict search ranking, traffic volume, or guaranteed outcomes.**"
   - Findings by severity (P0 → P1 → P2)
   - Backlog summary
   - Recommendations
5. **Export with checksum** — Markdown file + SHA-256 hash stored in export metadata

---

No-ranking Guardrail

Every audit (both local_readiness and live_crawl modes) stores a guardrails array in the audit JSON. The UI renders these unconditionally on every audit detail page.

live_crawl guardrails

```json
[
  "Live crawl audit only; findings reflect publicly observable signals at crawl time.",
  "No Google Search Console, Analytics, paid keyword APIs, or private CMS data is used.",
  "This audit does not predict search ranking, traffic volume, or guaranteed outcomes.",
  "Findings must not claim ranking or traffic impact.",
  "This is a basic public-page audit. It does not use Google Search Console, Analytics, paid keyword APIs, or private CMS data."
]
```

These are injected into the client report's **Scope section** and displayed on the audit detail page. The generator throws an error if any finding text contains forbidden claim words.

Deploy Path

Target environment: Azure App Service (Linux container), Sweden Central **Registry:** alairegistry.azurecr.io

Image tag: seo-readiness-portal:20260602-real-audit (date + purpose semantic tag)

Public URLs:

- https://seo-tools.alai.no/partners (Cloudflare Access authenticated)
- https://seo-tools.snowit.ba/ (custom hostname via MC #102750, Cloudflare TLS termination)

Origin protection: Azure App Service origin is IP-locked to Cloudflare ranges (403 on direct access to seo-readiness-alai.azurewebsites.net from non-Cloudflare IPs)

Deploy steps (manual operator path)

`bash
cd /Users/makinja/business/ALAI-Holding-AS/products/SEO-Readiness-Portal
`

1. Local gates (type-check, build, validate)

npm run type-check && npm run build && npm run validate:spec && npm run validate:phase12

2. Build image (ACR Tasks, remote build in Azure)

az acr build -r alairegistry -t seo-readiness-portal:20260602-real-audit .

3. Update App Service container config

az webapp config container set \ --resource-group rg-seo-readiness-prod \ --name seo-readiness-alai \ --container-image-name alairegistry.azurecr.io/seo-readiness-portal:20260602-real-audit \ --container-registry-url https://alairegistry.azurecr.io

4. Restart App Service

az webapp restart --resource-group rg-seo-readiness-prod --name seo-readiness-alai `

Post-deploy verification (ZAKON PI2 Check 4)

`bash

Confirm new image is active

az webapp config container show -g rg-seo-readiness-prod -n seo-readiness-alai \ --query "[?name=='DOCKER_CUSTOM_IMAGE_NAME'].value" -o tsv

Verify public endpoints (expect 302 CF Access redirect)

curl -sI https://seo-tools.alai.no/api/health curl -sI https://seo-tools.snowit.ba/api/health

Verify origin is IP-locked (expect 403)

curl -sI https://seo-readiness-alai.azurewebsites.net/api/health

Confirm Bilko domain untouched

dig +short bilko-demo.alai.no # expect ghs.googlehosted.com `

Final UAT (pending CEO/Proveo): Authenticated browser through Cloudflare Access → create client → run live audit → verify real findings from actual crawl → export report → confirm no-ranking disclaimer present.

Rollback

```bash
az webapp config container set \
  --resource-group rg-seo-readiness-prod \
  --name seo-readiness-alai \
  --container-image-name alairegistry.azurecr.io/seo-readiness-portal:20260531-cloud \
  --container-registry-url https://alairegistry.azurecr.io
```

az webapp restart --resource-group rg-seo-readiness-prod --name seo-readiness-alai

`

Previous known-good image: 20260531-cloud (pre-A1 local-readiness-only version)

Operator Runbook

How to run a live audit

1. **Authenticate:** Visit

https://seo-tools.alai.no/partners with Cloudflare Access credentials
2. **Create client:** Fill intake form (company name, website, services, competitors, Google access status)
3. **Trigger audit:** Click "Run Live Audit" on the client detail page
4. **Wait:** Audit takes 10–45 seconds (home + robots + sitemap fetches)
5. **Review findings:** Navigate to /clients/[clientId]/audits/[auditId] — see P0/P1/P2 findings with evidence JSON
6. **Convert to backlog:** Click "Add to Backlog" on any finding that needs operator action
7. **Generate report:** Click "Generate Report" → draft created with scope disclaimer + findings + backlog summary
8. **Export:** Click "Export Markdown" → .md file with SHA-256 checksum stored in workspace
9. **Handoff:** Fill checklist (client approved scope, evidence URLs verified, no forbidden claims) → generate handoff summary → generate partner follow-up package

How to deploy a new version

Follow the **Deploy steps** section above. Always run local gates before building the image. Always verify post-deploy (CF Access 302, origin 403, Bilko untouched).

How to rollback

Run the **Rollback** command. The previous known-good image is tracked in

DEPLOY-MAP.md. Verify rollback with the same post-deploy checks.

Troubleshooting

| Symptom | Likely cause | Fix | |---------|--------------|-----| | Audit hangs at "running" | SSRF timeout or AbortController not firing | Check Azure logs for timeout errors; verify

TOTAL_AUDIT_TIMEOUT_MS env var |
Audit returns empty findings	Site is behind Cloudflare challenge or 403 IP block	Expect P0 "crawl-blocked" finding; client must allowlist ALAI crawler UA or IP
"Response body exceeded 2 MB cap" error	Large home page or sitemap	Expected behavior; emit P1 finding "home page too large"
workspace.json corruption	Concurrent writes from multiple Azure instances	Restart App Service, restore from /home/data/workspace.json.backup- if present
Report contains forbidden claim words	Generator failed to catch; regex bypass	Report to John; update forbiddenClaimWords regex in generator.ts and runner.ts

---

Google Integration (Deferred)

Status: NOT IMPLEMENTED **Scope:** MC #102806 (B1 from REAL-AUDIT-ENGINE-PLAN-2026-06-02.md)
Requirements: Google Cloud OAuth client ID + secret, consent screen approval, token store (file or Postgres)
Blocked until: CEO provides/approves Google Cloud project + OAuth credentials

The current live crawl audit does **NOT** fetch Google Search Console impressions/clicks/queries or Google Analytics (GA4) page views/conversions. The

searchConsoleStatus and analyticsStatus fields in the intake form are **metadata-only** — they record the client's access status but do not connect to Google APIs.

When Google integration is implemented (follow-on MC), the audit will:

- Fetch impressions/clicks/queries from Search Console (last 90 days)
- Fetch page views/conversions from GA4 (last 90 days)
- Emit P0 findings if indexing errors are detected (e.g., "Discovered - currently not indexed")
- Emit P1 findings if query CTR < 2% for top-impression queries

The no-ranking-guarantee disclaimer will be updated to: "This report includes Google Search Console and Analytics data. Findings reflect historical performance only. **We do not guarantee future ranking, traffic volume, or conversion outcomes.**"

Technical Decisions Log

CEO decisions (2026-06-02, "sve preporu?eno, idi")

Decision	Rationale	Known limit	Follow-on
(a) File backend	Deterministic, testable, works for single-operator phase	Last write wins on concurrent access	Postgres migration MC
(b) Sync Server Action	MVP path, fits Azure 230s request ceiling	Max 45s total for 3 fetches; concurrent operators share slots	Async job queue MC
(c) Pure TS + cheerio	Lea Verou panel feedback: regex = hard no; cheerio handles broken HTML	None	None
(d) Existing audit detail route	Reuse /clients/[clientId]/audits/[auditId] — no new route	None	None
(e) Max one live audit in-flight per client	Enforced in runLiveCrawlForClient()	If operator triggers two audits rapidly, second is rejected	Queue or parallel-audit MC
(f) 403/CF challenge → P0 finding	Caller detects HTTP status, emits P0 "crawl-blocked"	No retry logic	Follow-on MC if retry needed

Correctness over Python parity

The TS implementation **fixes bugs** present in the Python reference (run-basic-seo-audit.py):

1. **Charset detection** — Python defaults to UTF-8 without checking Content-Type or; TS uses TextDecoder with sniffing
2. **og:image relative URL** — Python omits og:image entirely; TS detects relative URLs and emits P2 finding
3. **sitemapindex nesting** — Python silently ignores ; TS detects and emits P1 finding
4. **Canonical vs final URL** — Python compares canonical against requested URL; TS compares against response.url (after redirects)

Proveo verification outcome

All 3 child MCs (A1 #102801, A2 #102802, A3 #102803) were **independently verified by Proveo (Angie Jones)** after CodeCraft build:

- A1: type-check/build/validate EXIT 0, additive files intact, SSRF guard coverage confirmed
- A2: findings-to-backlog widening verified, evidence-URL done gate confirmed
- A3: **Bug caught in verification** —

forbiddenClaimWords regex threw on live_crawl scope text ("ranking", "guaranteed"). CodeCraft fixed + added validate:phase12 regression test. Proveo re-verified PASS.

Evidence:

/tmp/alai/996bd450/evidence-102800/verification.json, /tmp/alai/996bd450/evidence-102803/fix-verification.json

---

Open Items and Follow-on MCs

| Item | Priority | Description | Tracking | |------|----------|-------------|----------| | DNS-rebind SSRF guard | M | Runtime

dns.lookup check before fetch (currently only literal IPs blocked) | Follow-on MC |
Per-operator rate limiting	M	Prevent abuse: max 10 audits/hour per partner	Follow-on MC
Postgres migration	H	Replace file backend with Postgres for findings/backlog/audits	Follow-on MC
Async job queue	H	Move crawl to background worker (Redis/BullMQ) to unblock Server Action thread	Follow-on MC
Google Search Console integration	H (BLOCKED)	OAuth + impressions/clicks/queries (needs CEO-provided credentials)	MC #102806
Google Analytics (GA4) integration	M (BLOCKED)	OAuth + page views/conversions (needs CEO-provided credentials)	MC #102806

Playwright authenticated UAT	H	Browser through CF Access → run audit → verify findings (pending CEO login)	MC #102804
Retry logic for 403/503	L	Exponential backoff + retry on transient errors	Follow-on MC
Concurrent audit limit per partner	M	Allow 3 audits in-flight per partner (vs current 1 per client)	Follow-on MC

References

- **Plan:**

/Users/makinja/business/ALAI-Holding-AS/products/SEO-Readiness-Portal/REAL-AUDIT-ENGINE-PLAN-2026-06-02.md
BUILD-BLUEPRINT: /Users/makinja/business/ALAI-Holding-AS/products/SEO-Readiness-Portal/BUILD-BLUEPRINT.md
DEPLOY-MAP: /Users/makinja/business/ALAI-Holding-AS/products/SEO-Readiness-Portal/DEPLOY-MAP.md
Evidence: /tmp/alai/996bd450/evidence-102800/verification.json (A1/A2/A3 Proveo PASS), /tmp/alai/996bd450/evidence-102820/verification.json (deploy)
Python reference: ~/business/ALAI-Holding-AS/sales/seo-automation/run-basic-seo-audit.py (277 lines, public-URL crawl)
Validation script: scripts/validate-phase12.ts` (regression test for A3 fix)

Last updated: 2026-06-02 **Owner:** Skillforge (docs) / CodeCraft (implementation) / Proveo (verification) **Status:** DEPLOYED to production, pending authenticated browser UAT (MC #102804)

Revision #1
Created 2026-06-02 20:44:30 UTC by John
Updated 2026-06-02 20:44:30 UTC by John