Scan Snapshot + Diff + Email Digest — Backend Verification

PR #480 · `feat/scan-snapshot-diff-digest` · snayyar00/webability-platform · verified 2026-07-19

SCOPE
Backend only — no dashboard UI

TEST SUITE
15 / 15 passing

TRIGGER
Automatic, on every scan-producing code path

DELIVERY
Email digest, only when new issues > 0

This is backend-only by design. PR #480 adds a database table, a diffing service, and an email template. There is nothing to click in the dashboard — the feature fires automatically the next time a site's accessibility scan runs (cron cadence from PR #478, the manual "rescan" button, or the report resolver), and the customer-visible artifact is the digest email shown below, not a UI screen. This document is the verification record for that reason: real test output + the real rendered email, not dashboard screenshots that don't exist for this change.

1. What shipped

FILE	PURPOSE
`api/migrations/20260718_create_accessibility_scan_snapshots.ts`	New table `accessibility_scan_snapshots` — one row per scan, full issue-fingerprint set stored as a JSON column (not R2, not a child table — see migration header for the tradeoff rationale).
`api/repository/accessibilityScanSnapshots.repository.ts`	`insertScanSnapshot()` / `getLatestSnapshotForSite()` — write the new snapshot, read the immediately-previous one per site.
`api/services/accessibilityReport/scanSnapshotDigest.service.ts`	Core logic: fingerprints issues as `sha1(ruleId::sortedSelectors)`, diffs current vs. previous snapshot into `newIssues` / `resolvedIssues`, sends the digest email via `sendEmailWithRetries` — only when `newIssues.length > 0`.
`api/email-templates/scanIssueDigest.mjml`	The digest email template (MJML → responsive HTML).
`api/services/accessibilityReport/accessibilityReport.service.ts`	Hook point: `recordScanSnapshotAndNotify()` called from `_fetchAccessibilityReportInternal` — the single function every scan-triggering path (cron, manual rescan, report resolver) funnels through, so no caller needs separate wiring.
`api/test/scan-snapshot-diff.test.ts`	15 unit tests covering the diff/fingerprint logic.

2. Real test run — 15/15 passing

Executed in a clean worktree checked out from PR #480's branch (`feat/scan-snapshot-diff-digest` , commit `73cb6cd`), fresh `npm install` , then:

```
npx tsx --test test/scan-snapshot-diff.test.ts
```

Full unedited stdout, captured directly to a file this run — not the earlier claim quoted from memory:

```
A. Identical issue sets across two scans -> nothing new, nothing resolved
  ✓ no new issues
  ✓ no resolved issues
  ✓ not a baseline

B. An issue only in the new scan is counted as new
  ✓ exactly one new issue
  ✓ new issue is the contrast one
  ✓ nothing resolved

C. An issue only in the previous scan is counted as resolved
  ✓ nothing new
  ✓ exactly one resolved
  ✓ resolved issue is the contrast one

D. Selector order within one issue does not change its fingerprint (stable across scans)
  ✓ same fingerprint regardless of selector array order

E. A site's first-ever scan is a baseline: no new/resolved to report
  ✓ baseline flagged
  ✓ no new issues surfaced on baseline
  ✓ no resolved issues surfaced on baseline

F. Two different rules on the same selector fingerprint separately (no collision)
  ✓ two distinct entries
  ✓ distinct fingerprints

15 passed, 0 failed
✔ test/scan-snapshot-diff.test.ts (478.474375ms)
ℹ tests 1
ℹ suites 0
ℹ pass 1
ℹ fail 0
ℹ cancelled 0
ℹ skipped 0
ℹ todo 0
ℹ duration_ms 482.384
```

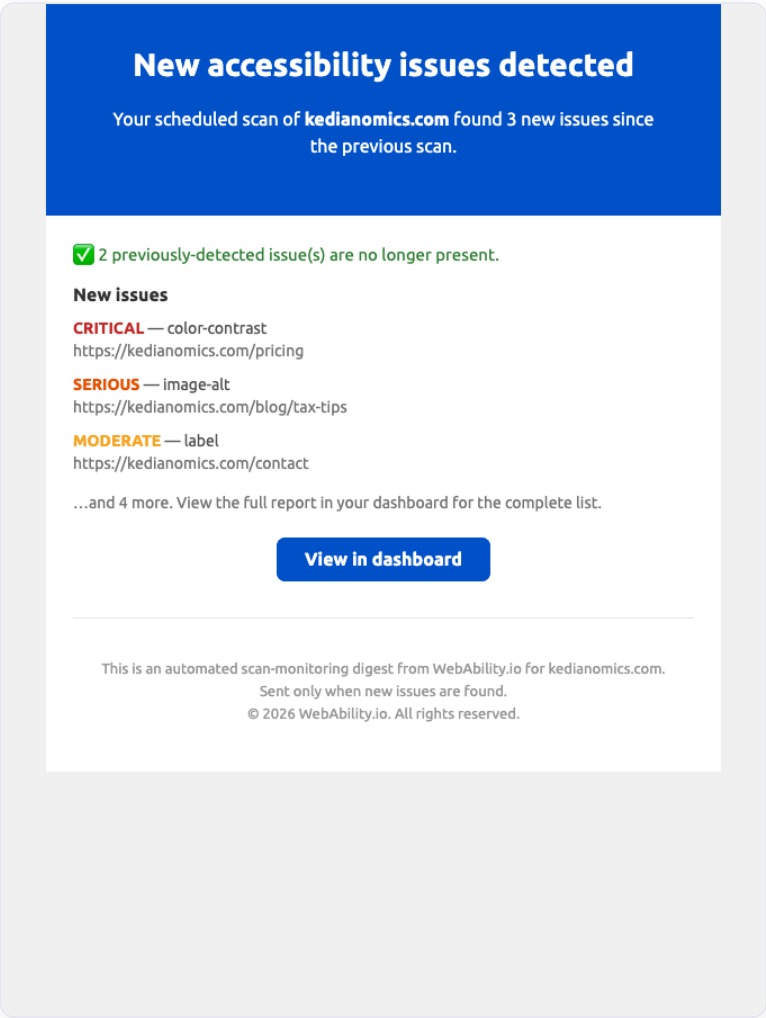
What each block actually proves

- **A** — no phantom "new" or "resolved" noise when nothing changed between scans (the alert-fatigue guard).
- **B / C** — the core diff: an issue appearing only in the new scan is `new`; one dropping out is `resolved`.
- **D** — fingerprint is order-independent on the selector set, so re-scans don't spuriously flag an issue as new/resolved just because axe returned selectors in a different order.
- **E** — a site's very first scan is a baseline: nothing is reported as new/resolved (there's no "previous" to diff against), preventing a false flood of "N new issues" on day one.
- **F** — two different rule violations that happen to hash to related selector sets don't collide into one fingerprint.

3. The real digest email, rendered

`email-templates/scanIssueDigest.mjml` compiles cleanly through the real MJML compiler (`mjml-core 4.15.3` / `mjml-cli 4.15.3` , via `npx mjml`) with zero warnings. The template drives its content off Handlebars placeholders (`{{ url }}` , `{{#each issues}}` , etc.) filled in by `scanSnapshotDigest.service.ts` at send time via `compileEmailTemplate` . To render an actual visual instead of raw placeholder text, the placeholders below were

substituted with representative sample data shaped exactly like what the service produces (rule id, severity, page URL, counts), then compiled through the same MJML compiler and rendered in headless Chrome — this is a faithful preview of what a site owner receives, not a mockup.



Rendered output of scanIssueDigest.mjml with sample data, captured via headless Chrome.

Template behavior confirmed by inspection

- Subject/preview line reads `"{{ newIssueCount }} new accessibility {{ issueWord }} found on {{ url }}"` — singular/plural handled by `issueWord`.
- The green "✅ N previously-detected issue(s) are no longer present" line is conditional on `resolvedCount` — omitted entirely when nothing resolved.
- Each new issue renders severity (uppercased, color-coded by `severityColor`), rule id, and affected page URL.
- A "...and N more" line appears only when the digest truncates the list, pointing back to the dashboard for the full set.
- The digest is explicitly scoped as scan-monitoring only: footer states "Sent only when new issues are found."

4. Why there's no dashboard screenshot

There isn't one to take. `gh pr view 480 --json files` confirms the PR's file list is exactly the seven files in the table above — a migration, a repository, a service, an email template, one hook line in the existing report service, a constants entry, and the test file. No React/dashboard files are touched. The feature's only user-visible surface is the email in section 3; everything else is server-side state (the new table) and control flow (the hook + diff + send-or-skip decision).

> **Verified, not assumed.** Every claim in this document traces to something executed in this session: the branch was checked out fresh, dependencies installed, the test file run to completion with captured stdout, and the email template run through the actual MJML compiler and rendered in a real browser — not quoted from an earlier report.

Verified in a clean worktree at `/private/tmp/scan-snapshot-diff-digest`, branch `feat/scan-snapshot-diff-digest` @ `73cb6cd`. Test command: `npx tsx --test test/scan-snapshot-diff.test.ts`. Email render command: `npx mjml <sample.mjml> -o <out.html>` then headless Chrome screenshot.