Daily site reports and project management for a construction company

Jake Tae-moon Jo · Engineering lead · https://jake-jo-portfolio.vercel.app/en/work/construction-pmis/

Started from a general construction project management system (PMIS), rebuilt it around the client's paper daily reports, cut finance features already built, and took it to client acceptance testing (UAT).

Client UAT	1 day	2 scope cuts	12
acceptance testing reached	redesign to report-core merge	of features already built	manuals: 4 roles × 3 languages

Results

It began as a general-purpose PMIS, was rebuilt around the daily site report, and went as far as client UAT.

Every change request from the client was shipped within 24 hours.

Built a site PMIS centred on the daily report, as a web app in 3 languages. One flow covers designating the reporter, writing, submission checks, site-manager approval, director edits after approval, the daily-report PDF and the roll-up of approved rows.

Put staging on the client's domain on Jul 8, and took the system as far as client acceptance testing.

Merged the 7-phase daily-report core the day after the redesign around two of the client's paper reports (May 10).

Took the schedule/EVM and finance features already built out of scope, fitting the product to the daily report and the physical flow of materials.

Produced 12 user manuals, for 4 roles in 3 languages. Their 450 screenshots are re-shot by a capture script.

Period

Apr – Jul 2026

Role

Only human on the build team; designed and ran the AI agent team

Team

Me plus an AI agent team (11 roles defined)

Platforms

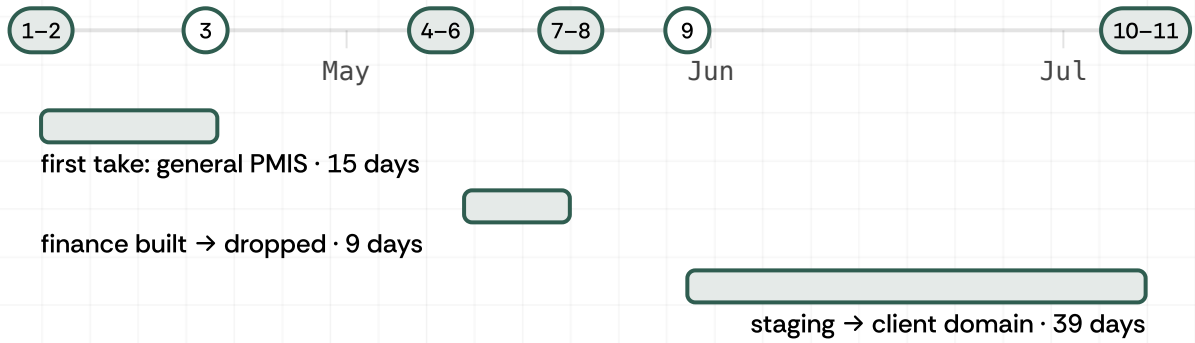
Web app (per-role screens for head office and sites, 3 languages), daily-report PDF

Context

A general contractor. Two of the client's paper daily reports served as the template for the input screens

From a general PMIS to client UAT

It began as a general-purpose PMIS, was rebuilt around the daily report and shed finance, and went as far as client UAT.

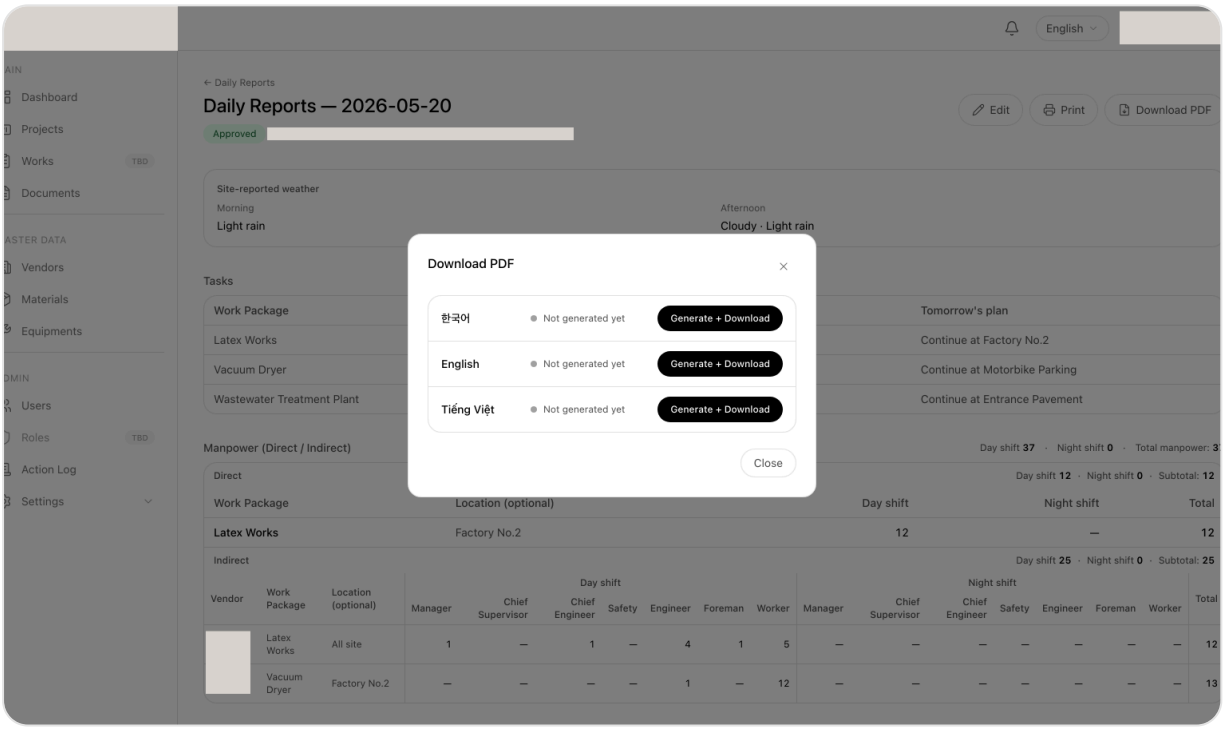


1. Apr 5, 2026 — **First commit**
   Started with a 7-role agent team
2. Apr 7, 2026 — **First deploy**
   Up to document management and access control
3. Apr 19, 2026 — **Schedule analytics**
   Status lights, S-curve, critical path (through Apr 20)
4. May 9, 2026 — **Redesigned around the daily report**
   Built on two of the client's paper reports; schedule and EVM archived
5. May 10, 2026 — **Daily-report core merged**
   Seven phases; first manual
6. May 11, 2026 — **Finance built**
   Orders, receipts, payments, cost (through May 12)
7. May 20, 2026 — **Finance taken out of scope**
   "Orders, payments and cost belong to the external accounting system"; three material catalogues
8. May 24, 2026 — **Manuals rewritten**
   4 roles × 3 languages = 12

9	May 30, 2026	**Staging built** Incremental deploy script (through May 31)
10	Jul 8, 2026	**Staging on the client's domain** Certificate applied
11	Jul 9, 2026	**Email notifications; last commit**

Screens

PDF download dialog on a daily report. Pick one of three languages (Korean, English, Vietnamese), generate, then download. Seed data; logo and vendor names masked.



Problem and context

The job was to turn the client's paper daily report into a web flow where one designated person per project writes it and the site manager approves it.

The client is a general contractor. The users are head-office admins, directors, site managers and staff. The product centres on the daily site report. Each project has one designated reporter who writes one report a day, and the site manager or a director reviews and approves it. A report covers the work done, direct labour, subcontractor labour (7 trades × day and night), equipment, materials used, photos, drawings and other attachments. The subcontractor, equipment and material masters, the notifications and the photo and drawing uploads all exist to support this flow.

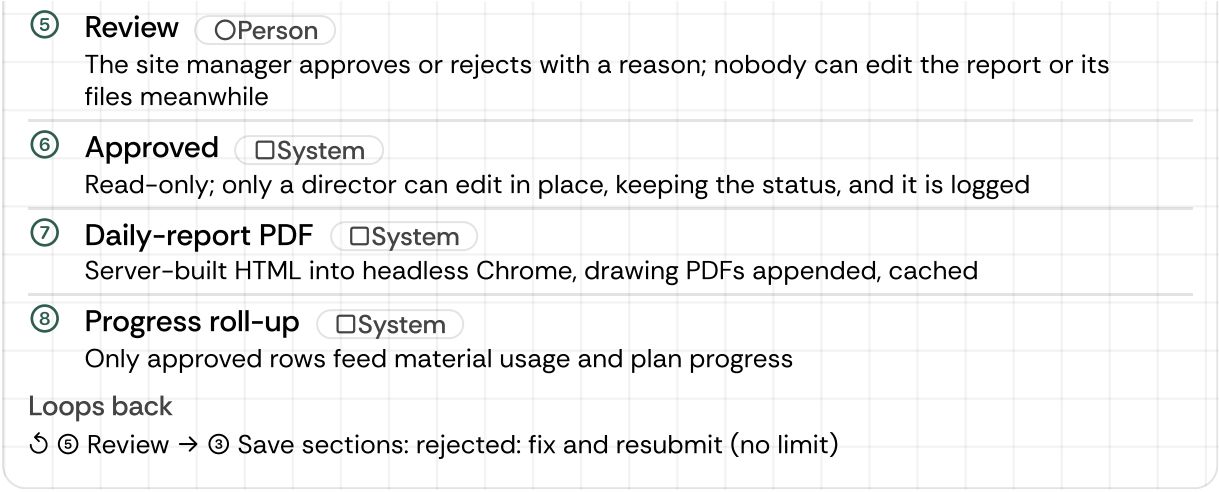
It did not start this way. In April it was a general-purpose PMIS with a task WBS and Gantt chart, document approval and schedule performance analysis (EVM). On May 9 it was redesigned around the daily report, using two of the client's paper daily reports as the template, and on May 20 the finance features already built were taken out of scope (key decisions 1 and 2).

Permissions were the hard part. The same staff member writes reports on one project and only reads them on another. The site manager differs from project to project as well. An approved report is both the unit of audit and the basis for progress figures, so who may change what after approval had to be settled too. The screens had to come in 3 languages, and notifications had to go out in each recipient's language.

One daily report, end to end

Each project has one designated reporter who writes one daily report a day; the site manager approves it, and only approved rows count toward progress.

1. **Assign the reporter** ○Person
   The director sets the site manager and members and names one reporter (the DB allows one active)
2. **Today's draft** ○Person
   One per project and day; yesterday's plan for tomorrow pre-fills today's work
3. **Save sections** □System
   Saving is never blocked; each row: lock check, change, audit, cache invalidation; other projects' ids rejected
4. **Submit** ○Person
   The editor checks 11 rules first, the server re-checks 6; weather attached after submit



My role

I was the only human; I defined the agent team from the first commit and ran it, and I decided the design direction and the merges.

I was the only human on the build team. I am the only git author, and 177 of the 290 commits (61%) carry an AI co-author trailer; leaving out merge commits, it is 177 of 239 (74%). Commits cannot separate what I wrote by hand from what the agents wrote, so the role reads "Only human on the build team; designed and ran the AI agent team".

The agent team was in the repo as definition files from the first commit (Apr 5). It started with 7 roles: orchestrator, designer, frontend, backend, reviewer, API docs and QA. TDD guide, git flow (Apr 19), manuals (May 10) and devops (May 22) followed, for 11 roles. Orchestrator, backend, frontend and devops were set to Opus; the other 7 to Sonnet. The definitions differ from the public templates and were written for this project. The pipeline was written into the instructions as: start branch → design → failing test first → implement → tests pass (80% coverage) → review and API docs → QA (e2e) → manuals → PR.

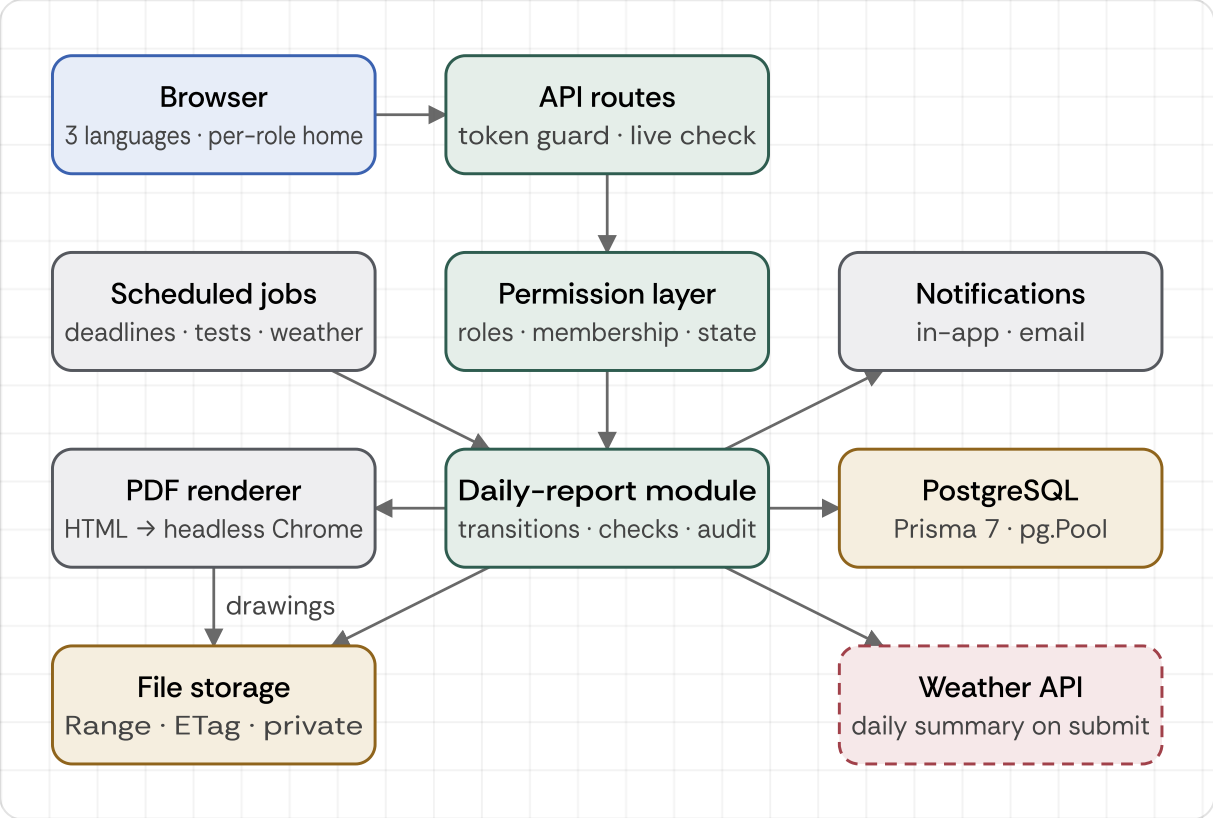
There were 110 PRs, and the 106 that were merged were all merged from my account. No person reviewed a PR. Some decisions are visibly a person's in the record. On May 12 I closed a PR that moved the UI onto the IBM Carbon design system, noting that it was "not the design direction this product is shipping". The design went to its own grayscale

system, with shadcn ruled out. On May 14 I prepared a proposal for a client meeting (a workforce-management module, 16 pages) asking for sign-off on the screen designs. That module was never built.
The decision log holds 42 entries, 25 of which list alternatives. As its header says, these are decisions the agent made with its recommended defaults and then carried out. The scope-change design document lists 8 more decisions confirmed on May 20. The alternatives and reasons on the decision cards come from these two documents and from commit messages.

System architecture

System architecture

Changes to a daily report's rows go through one recipe in the daily-report module: lock check, change, audit log, PDF cache invalidation.



Browser

3 languages (1,600 keys each); every time is shown in one fixed local time zone. Each role lands on a different home: directors on a portfolio dashboard, site managers on a review queue, staff on project tiles.

API routes

Home-grown JWT. Access tokens last 120 minutes, refresh tokens 7 days (httpOnly cookie + DB). Refresh tokens rotate on every use, and a detected reuse ends all of that user's sessions. The auth guard re-checks in the DB on every request that the user is still active.

Scheduled jobs

Submission-deadline reminders (reporter and site manager), a test reminder 28 days after concrete is used, and an hourly weather backfill.

Permission layer

Four base roles + project membership + one site manager per project + the designated reporter (a designation kept apart from the four roles; a partial unique index in the DB allows one active) + locks that follow the report's state. Whether a document's edit button is disabled is decided by the server and sent in the detail response.

Notifications

All notifications go through one service and are rendered in the recipient's language when written. Polling pauses while the page is hidden. Email and a signed-token web view were added on Jul 9.

PDF renderer

One page per report. The server builds HTML, feeds it to headless Chrome and appends the drawing PDFs. The cache is cleared from 18 mutation paths; concurrent renders are merged into one and cut off after 30 seconds.

Daily-report module

State transitions are one pure function. Saving is never blocked; only submission is checked (6 rules enforced by the server). Row changes go through lock check, change, audit log and PDF cache invalidation, and another project's ids are rejected at save time.

PostgreSQL

PostgreSQL + Prisma 7. The audit log never blocks a response when it fails, and edits and deletes are stored as per-attribute diffs.

File storage

Files sit behind a storage abstraction. Downloads stream, with Range, a weak ETag and a private cache, so that a shared CDN never serves one user's bytes to another.

Weather API

When a submission succeeds, 14 daily-summary values and derived indicators (including a concrete curing index) are stored. A failure never blocks the submission; there is a retry button and a backfill.

`prisma/migrations/…_major_reporter_active_unique/migration.sql` · One active reporter per project is enforced in the DB itself, by a partial unique index. An ended designation has effectiveTo set and drops out of the index. (The whole migration.)

```
-- Ensure at most one active MajorReporter per project
-- (effectiveTo IS NULL).
CREATE UNIQUE INDEX "MajorReporter_projectId_active_key"
  ON "MajorReporter"("projectId")
  WHERE "effectiveTo" IS NULL;
```

`src/lib/auth/document-permissions.ts` · A document attached to a report follows the report's state: after approval only directors and system admins may edit it, under review nobody may, and in draft or rejected the active reporter may too. (Comments condensed from the source; the source builds each error object inline, folded here into an illustrative deny.)

```
// Documents attached to a daily report follow the report's state:
//   DRAFT / REJECTED → active Major Reporter or SYSTEM_ADMIN / DIRECTOR
//   SUBMITTED        → nobody (must be reviewed first)
//   APPROVED         → SYSTEM_ADMIN / DIRECTOR only
export async function gateDocumentMutation(
  caller: Caller,
  doc: DocumentForPermission,
): Promise<MutationGate> {
  const report = await findAttachedReport(doc.id);
  if (!report) {
    if (await canEditDocument(caller, doc)) return { ok: true };
    return deny(403, 'FORBIDDEN');
  }
  const isAdmin = canSeeAllProjects(caller.role);
  if (report.status === 'APPROVED') {
    if (isAdmin) return { ok: true };
    return deny(403, 'REPORT_APPROVED_DIRECTOR_ONLY');
  }
  if (report.status === 'SUBMITTED') {
    return deny(400, 'REPORT_UNDER_REVIEW');
  }
  // DRAFT or REJECTED
  if (isAdmin) return { ok: true };
  const isReporter =
    await isActiveReporter(caller.userId, report.projectId);
  if (isReporter) return { ok: true };
  return deny(403, 'FORBIDDEN');
}
```

`src/lib/daily-report/mutate.ts` · The routes that change a daily report's child rows go through this one function: edit lock, change, edit time if approved, then PDF cache invalidation and the audit log. (Comments condensed from the source; the refusal mapping is folded into an illustrative refuse and the audit-log arguments are written out.)

```
/**
 * Centralises the four cross-cutting concerns of a daily-report
 * child mutation:
 *   1. Edit gate    — status + actor checks
 *   2. Mutation     — the caller's callback, against the gated report
 *   3. lastEditedAt — bumped iff the report was APPROVED at gate time
 *   4. Side-effects — PDF cache invalidation + audit-log entry
 */
export async function performReportChildMutation<T>(
  input: PerformChildMutationInput<T>,
): Promise<PerformChildMutationOutcome<T>> {
  const { reportId, request, caller, audit, mutate } = input;
  const gate =
    await gateReportChildEdit(reportId, caller.userId, caller.role);
  if (!gate.ok) return refuse(gate);
  const report = gate.report!;
  const result = await mutate({ report });
  if (report.status === 'APPROVED') {
    await prisma.dailyReport.update({
      where: { id: reportId },
      data: { lastEditedAt: new Date(), lastEditedById: caller.userId },
    });
  }
  invalidateDailyReportPdfCache(reportId).catch(() => {});
  const d = typeof audit === 'function' ? audit(result) : audit;
  logAction({ userId: caller.userId, ...d, request });
  return { ok: true, result };
}
```

Key decisions

Two of the six cut features already built; four keep the client's real report and the approved record, three of them on agent defaults.

From a schedule- and EVM-centred general PMIS to one built around the

daily report

Problem

What existed in April was a general-purpose PMIS with a task WBS and Gantt chart, document approval and schedule performance analysis (status lights, S-curve, critical path). Then two of the client's paper daily reports came in as design input. The client's form had sections for work, labour, subcontractor labour, equipment, materials, photos and drawings.

Options considered

Auto-collect plus manual notes: gather the day's changes from the activity log and let the site manager add weather and remarks (option C in the Apr 7 plan)

✓ Follow the client's form: the designated reporter fills in each section and the site manager approves

Decision

The design followed the client's form. Each project has one designated reporter who writes one report a day, and the site manager or a director approves it. The subcontractor, equipment and material masters, notifications and photo and drawing uploads were all repositioned to support that flow. The schema was rewritten and the migration history discarded; schedules, tasks, labour, attendance and document approval moved to an archive folder. Why option C was dropped is not on record; the plan only says it was "superseded by SPEC.md".

Result

The next day (May 10) daily-report phases 1–7 were merged (PRs #12–#19). EVM was kept in the archive as a "future visualization phase".

Cost & remaining risk

The schedule analytics and task management built in April left the product.

Take the finance features already built out of scope, and fix materials to three catalogues

Problem

On May 11–12 purchase orders (three-step approval), receipts (with evidence), payment requests and a cost summary (Excel and PDF) were built. That left the question of whether a site PMIS should carry the flow of money or only the flow of materials.

Options considered

Keep finance in the PMIS (the orders, receipts, payments and cost built on May 11–12)

✓ Archive finance; the site PMIS handles only the physical flow of materials

Decision

All finance menus and models were archived. The design document records two reasons: "orders, payments and cost are handled separately in the external accounting system (not PMIS scope)" and "fixing materials to three kinds cuts the data-entry load and simplifies testing and tracking". The site PMIS would focus on the flow of materials: supplier → truck → site location → use → test. Materials became three fixed catalogues (concrete, rebar and stone), the daily report got a single "usage (= receipt)" event, and tasks got a two-level location. The 8 decisions confirmed that day also record what was deferred: locations stop at two levels (three or more later), concrete tests are only marked complete (structured strength and slump later), and notifications stay in the app (email and SMS later).

Result

It shipped on May 20 as 9 PRs (#57–#65). For concrete, a scheduled job sends the test reminder 28 days after the day of use. The deferred email notifications were added on Jul 9.

Cost & remaining risk

Twelve finance UI routes, ten API routes and their tables were removed. Per the work log, the passing unit-test count fell from 1,855 (May 18) to 790 (May 20).

Permissions from roles, membership and the designated reporter; attachment locks from the report's state

Agent default · no sign-off on record

Problem

The same staff member writes reports on one project and only reads them on another. The site manager differs from project to project too. And the photos, drawings and documents attached to a report were also documents in their own right that could be opened and edited from the document list.

Options considered

Use only the documents' existing creator and role rules

Add a separate "freeze" flag

✓ Follow the report's state: in draft or rejected, the reporter plus directors and system admins; under review, nobody; after approval, directors and system admins only

Decision

There are only four roles (system admin, director, manager, staff); everything else is settled inside the project: membership, one site manager per project, and the designated reporter. The designated reporter is a designation kept apart from the four roles, and a partial unique index in the DB keeps it to one active per project (code below). Edit rights on documents attached to a report were tied to the report's state, because, as the decision log puts it, "the daily report is the audit unit". Evidence must not change while a

report is under review, and after approval only directors and system admins may touch it, the same rule as for the report body. On May 18 the permission checks scattered across routes were gathered into named verbs such as canManageProjectMembers (5 paths).

Result

While a report is under review, nobody can edit its photos, drawings or documents (REPORT_UNDER_REVIEW); after approval, only directors and system admins can. Documents not attached to a report keep their usual document rules.

Cost & remaining risk

One site manager per project is held by an application check (409); the reporter rule is held by the DB. Permission checks also come in two styles: the subcontractor and equipment masters use table-based RBAC, while projects and reports use membership predicates.

The server decides the lock; the screen only receives the verdict

Agent default · no sign-off on record

Problem

The document detail screen had to disable the edit button on locked documents. But the screen's guess and the server's actual decision could disagree.

Options considered

The screen recomputes the lock from role, creator and attachment type

✓ Put the result of the server's gate function (canMutate, lockedCode) into the detail response

Decision

The server's verdict goes into the response. The decision log gives the reason: "the client heuristic missed the active-reporter check, so the disabled state diverged from the server". The screen and the API now use the same gate function. In the same spirit, permission overrides are not put in the JWT but read from the DB on every request, so an admin can revoke them without forcing a token re-issue.

Result

The document detail response carries canMutate and lockedCode, and the screen sets the button state from them. In the same direction, on May 18 the submission rules were moved into one module shared by the server and the editor. Of the 11 rules the server enforces 6; a missing weather entry is checked only in the editor.

Equipment quantities as decimals to two places

Agent default · no sign-off on record

Problem

The client's sample reports recorded equipment quantities such as "0.5 day". The first schema took the day and night equipment counts as integers.

Options considered

Keep integers and have reporters round

✓ Switch to Decimal(12,2)

Decision

They became decimals. The decision log's reason: "integer-only forced reporters to fake the data". On May 21 equipment time, until then entered in three blocks (morning, afternoon, extra), became a single start and end time (24-hour, past midnight allowed). The morning, lunch (excluded), afternoon and extra segments and the work-day count (hours worked ÷ 7) are computed for display, and existing rows were filled in by a migration.

Result

Each equipment row in a report now takes a plate number and one start and end time per machine; the segments and work days are computed on screen.

Daily-report PDFs: the server builds HTML from data and renders it in headless Chrome

Problem

Every report needed a one-page PDF with the attached drawing PDFs appended. The cost-summary PDF built in the same period used browser printing.

Options considered

Browser printing (the cost-summary PDF's approach): no new runtime dependency

Open the print-view URL in headless Chrome and capture it

✓ The server builds HTML from DB data and feeds it straight to headless Chrome

Decision

Rendering moved to the server (May 12). puppeteer-core drives headless Chrome, and pdf-lib appends the drawing PDFs. The first version navigated to the print view, but the headless browser had no login context, so it captured the login page. The next day it switched to building HTML on the server from DB data and feeding it in directly. Finished PDFs are cached as files, and 18 paths that change a report clear the cache. Concurrent renders of the same report are merged into one and cut off after 30 seconds.

Result

A download is served straight from the cache when there is one and rendered on the spot when there is not. One report becomes one PDF file, drawings included.

Cost & remaining risk

It brought in the runtime dependency the browser-printing option avoided. On May 13 the Chrome files went missing from the deploy bundle and text broke for lack of a font; file-tracing settings and a bundled font fixed both.

Quality and verification

A work log records 1,001 tests passing, but the test rules lived only in agent instructions and the nightly QA swallowed failures.

Counted statically, there are 45 unit test files (542 test blocks), 28 integration (API) files (255), 23 component files (190) and 45 e2e specs (429). The e2e suite is split into desktop and mobile projects. I did not re-run any of them for this write-up.

Most run results are work logs in commit messages. One May 25 commit records 1,001 unit, integration and component tests passing. Another commit the same day records 358 desktop e2e tests passing, 4 flaky, 54 skipped and 0 failing. A May 21 commit puts e2e coverage at about 85%: 28 of 33 use cases fully covered and 5 partly (per the work log).

On May 24, 86 stale e2e failures were cleaned up, and 43 tests (12 specs) that need rewriting were left skipped.

One run left a record file of its own: on May 21, all 6 smoke tests passed against a demo deploy running on seed data. From May 20 a nightly GitHub Actions job also ran per-role screen QA against that deploy. Its QA step, however, swallows a non-zero exit, so a "successful" run does not mean QA passed.

The pipeline rules (test first, 80% coverage, commit only after the reviewer, QA and manuals pass) lived only in the instructions. There were

no agent hooks and no CI running on PRs.

Two reviews are on record. On May 18 candidates from an architecture review run as an agent command became 5 refactoring PRs (#47–#51); sharing the submission rules between server and editor and gathering permission checks into named verbs both came from there. On May 25, 6 findings from a design review were applied.

One feature, start to finish

This follows one daily report from the reporter's assignment to the PDF and the progress roll-up.

1. **Assignment — director**

   A director creates the project, names the site manager, adds staff and designates the reporter. A new designation closes the previous active record and notifies both the previous and the new reporter.

2. **Draft — designated reporter**

   The reporter creates today's draft. There is one report per project and date. Yesterday's "plan for tomorrow" pre-fills today's "work done".

3. **Save**

   Saving is never blocked. Row changes go through a shared recipe: check the edit lock, apply the change, record the edit time if the report was approved, clear the PDF cache and write the audit log. Trade, location or subcontractor ids from another project are rejected at this step.

4. **Submit**

   The editor blocks first with the 11 shared rules, and the server re-checks 6 of them. On success the status becomes SUBMITTED and the site manager is notified in their own language. After the response is sent, the day's weather summary is fetched and attached.

5. **Locked under review**

   Until the review ends, nobody can edit the photos, drawings or documents attached to the report (REPORT_UNDER_REVIEW).

6 **Rejection and resubmission**

A site manager must give a reason to reject. The reporter's editor shows that reason, and the submit button turns into "Resubmit". There is no limit on resubmissions.

7 **Approval**

An approved report is read-only. Only a director can edit it in place; the status stays the same and no notification goes out. The edit time and the audit log are still recorded.

8 **PDF**

If a cached copy exists it is served at once. Otherwise the server builds the HTML, renders it to PDF in headless Chrome, appends the drawing PDFs and caches the result.

9 **Progress roll-up**

Only approved rows are added to the project's material usage and to the actuals on the work-plan timeline.

10 **Scheduled reminders**

If nothing is submitted by the deadline, a scheduled job reminds the reporter and the site manager. 28 days after concrete is used, a test reminder goes out.

Retrospective

Features already built were cut from scope twice, and every recorded incident has its cause and its fix on file.

Features were cut from scope after they were built, twice. Schedule and EVM were built on Apr 19–20 and archived on May 9; finance was built on May 11–12 and archived on May 20. When finance was planned, I handed it over with "do it all; I'll check at the end". Across the two cuts, 251 files went to the archive.

The documentation once lagged the code. The manual capture script passed over 404 and 403 pages without an error, so the manuals kept pointing at screens for features removed in the daily-report redesign. On May 24 the script was changed to stop at the first failure, and all 12 manuals were rewritten from scratch.

A refactor once removed a check. On May 10, when the trade on a task row became a selectable item, the only check that stopped another project's trade, location or subcontractor ids from going into a report row disappeared. On May 25 a project-scope check went in at save time, covering 7 routes.

The decision log's header says decisions would be revisited and adjusted after phase 7, but there is no record of that review. The gates were only instructions, too. The payroll-advance platform I started afterwards added hooks, a security reviewer, story IDs and decision markers.

Incidents and what changed

Every recorded incident has its cause and its fix on file; five deploy and PDF incidents were fixed on May 13 alone.

5
deploy and PDF incidents fixed on May 13 alone



1	Apr 7, 2026	**Type errors only in the build** Local dev skipped type checks → prisma generate in the build, one type path
2	May 13, 2026	**Five deploy and PDF incidents** Connection limit, demo deploys with an empty DB, missing Chrome files, PDFs of the login page, missing fonts → pool size, an explicit seed that stops at zero users, file tracing, server-built HTML, bundled fonts
3	May 22, 2026	**Document upload page 404** Restored a page lost in the redesign
4	May 24, 2026	**Manuals showed removed features** The capture script swallowed 404s → fail fast; all 12 manuals rewritten
5	May 25, 2026	**Rows could take another project's ids** A check lost in the May 10 refactor → project-scope check at save time (7 routes)
6	May 30, 2026	**Eleven pitfalls on the first staging build** Folded into runbook v0.3

How I work

This project was also built with a role-based AI agent team. The team and its gates differ from project to project; this one's are under "My role" above. Below is the method as written up while building the EWA platform.

See how I set up and ran the work on the 2026 EWA platform →

https://jake-jo-portfolio.vercel.app/en/process/

Contact

whxoans@gmail.com
+82 10-9233-7440