Gaps and Next Steps

Source: `~/ALAI/products/Plock/docs/demo-readiness/03-gaps-and-next-steps.md`

PLOCK — Gaps and Next Steps

Date: 2026-03-14
Purpose: Convert the current PLOCK reality into an actionable plan: what is missing, what is conflicting, what is risky, and in which order the remaining work should happen.

---

1. Goal

The original goal was to get PLOCK into a state where we can:

1. understand the real product and architecture,
2. produce deterministic documentation,
3. QA that documentation,
4. then generate implementation tasks by feature area,
5. and move toward demo-readiness / production-readiness in a controlled way.

This document defines what blocks that goal today and what sequence should be followed next.

---

2. Current State Summary

PLOCK already has:

- substantial real code,
- real integrations,
- real tests,
- real infrastructure scaffolding,
- real product and GTM documentation.

However, PLOCK is currently blocked by:

- conflicting architecture documentation,

- missing canonical documentation package,
- weak documentation-to-artifact execution flow,
- and lack of a clean, approved “source-of-truth → QA → backlog” chain.

---

3. Gap Categories

3.1 Canonical Documentation Gaps

GAP-DOC-001 — Canonical `docs/demo-readiness/` package exists, but is still incomplete

Severity: P0

The canonical demo-readiness package now exists, but it is still only a baseline layer.

Already present:

- source-of-truth baseline
- user stories baseline
- feature baseline
- gaps/next-steps baseline
- QA checklist baseline

Still missing or incomplete for the fuller package:

- program map
- deterministic task contract
- doc inventory gap audit
- package-level review expansion
- later demo/sales/support readiness docs
- program map
- deterministic task contract
- doc inventory gap audit
- user stories baseline
- feature catalog
- QA gate checklist

Impact:
Without this package, there is no stable documentation handoff point for QA or implementation backlog generation.

GAP-DOC-002 — No canonical feature catalog

Severity: P0

Features are spread across:

- code structure,
- PRD,
- GTM docs,
- API spec,
- runbooks,
- integration docs.

But there is no single authoritative feature catalog that says:

- what exists,
- what is partial,
- what is planned.

Impact:
Implementation task generation will drift or duplicate work.

GAP-DOC-003 — User stories are present, but not operationalized

Severity: P1

User stories exist mainly in `docs/PRD.md`, but not yet in a clean, implementation-ready canonical user-story document.

Impact:
Traceability between product vision and implementation work is weaker than it should be.

GAP-DOC-004 — Canonical QA checklist exists, but QA execution is still incomplete

Severity: P0

A trusted QA checklist now exists, but the review process is still immature and must continue with explicit review records and package verdicts.

What now exists:

- deterministic QA checklist baseline
- package review workflow

What still needs strengthening:

- continued document-by-document review records as the package grows
- repeatable reviewer discipline for future generated docs
- explicit re-review after any canonical doc changes

Impact:
QA can now start approving or rejecting the documentation layer, but the process still depends on careful manual review.

3.2 Architecture / Source-of-Truth Gaps

GAP-ARCH-001 — Conflicting architecture documents

Severity: P0

`docs/ARCHITECTURE.md` conflicts with the actual repo and stronger docs.

Examples of stale/conflicting claims:

- Next.js instead of MFE shell
- Express instead of Ktor
- Prisma instead of Exposed + Flyway
- `organization_id` instead of `warehouse_id`

Impact:
Any agent or human using the wrong document can generate incorrect output.

GAP-ARCH-002 — API spec drift

Severity: P1

`docs/API-SPEC.md` appears useful, but likely drifted from the actual backend implementation and tenant model.

Impact:
API-facing work may be built or reviewed against incorrect assumptions.

GAP-ARCH-003 — Runbook drift

Severity: P1

Different docs disagree on:

- npm vs pnpm
- startup flow
- JDK/runtime expectations

Impact:
Ops/docs/support/demo execution can become inconsistent and brittle.

3.3 Product / Feature Gaps

GAP-FEAT-001 — AI feature maturity unclear

Severity: P1

AI Chat and Smart Picking are core product promises, but actual implementation depth is not yet cleanly verified against code and runtime behavior.

Impact:
Demo promises may exceed current product maturity.

GAP-FEAT-002 — 3PL capability is more strategy than current product

Severity: P1

3PL/client portal/billing appears strongly in docs and GTM, but current implementation evidence suggests this is more Phase 2 than current MVP.

Impact:
Sales/demo scope can overpromise if not constrained.

GAP-FEAT-003 — Sales/marketing/support artifacts not packaged

Severity: P2

The content exists in fragments:

- GTM strategy
- pricing
- regulatory notes
- runbooks

But not yet as:

- demo script,
- objection handling pack,
- support playbook,
- onboarding checklist,
- sales-facing feature matrix.

Impact:
Cross-functional readiness is incomplete.

3.4 Security / Hygiene Gaps

GAP-SEC-001 — Secret-like material in infra docs

Severity: P0/P1

`infrastructure/README.md` contains environment variable examples that resemble real secrets or secret material.

Impact:
Security hygiene issue. Must be reviewed and cleaned immediately.

GAP-SEC-002 — Secrets/process docs need explicit verification pass

Severity: P1

Secrets strategy exists, but should be validated against:

- actual deployment practice,
- Vaultwarden usage,
- Railway variable setup,
- non-committed local env discipline.

Impact:
Production readiness risk if docs and reality differ.

3.5 Process / Orchestration Gaps

GAP-PROC-001 — Delegated doc output quality remains unreliable

Severity: P0

Even after improving task gating, agents still often:

- narrate output,
- claim files were created,
- fail to produce actual artifacts.

Impact:
Important documentation work cannot yet be trusted blindly.

GAP-PROC-002 — Contradictory orchestration logging still exists

Severity: P1

Observed behavior:

- task blocked by QA gate,
- then orchestrator still logs it as completed.

Impact:
Operational trust and status reporting are still imperfect.

GAP-PROC-003 — Forge worker pool saturation

Severity: P1

Some tasks are delayed or requeued because:

- forge pool is full,
- H-priority tasks compete with unrelated workload.

Impact:
Even correctly defined tasks may not progress quickly.

---

4. What Must Happen Before Feature-by-Feature Implementation Tasks

Before generating backend/frontend/db/integration/security/sales/marketing task waves, we need:

Required Preconditions

- canonical source-of-truth note
- canonical user stories baseline
- canonical feature baseline
- canonical gaps/next-steps note
- canonical QA checklist
- clear exclusion of stale architecture doc from planning
- security hygiene review of infra secret examples

If these are skipped, implementation tasks will be generated against mixed truths.

---

5. Recommended Execution Order

Phase 1 — Documentation Truth Lock

Priority: P0

Deliverables

1. Canonical source-of-truth note
2. User stories baseline
3. Feature baseline
4. Gaps and next steps
5. QA checklist baseline

Outcome:
One trusted planning layer.

Phase 2 — Documentation QA

Priority: P0

QA should validate the documentation set against:

- actual repo structure
- actual code domains
- actual migrations
- actual integrations
- actual security/tenant model

QA must explicitly check:

- no stale architecture assumptions
- no references to nonexistent files
- no contradictory stack descriptions
- no unsupported feature claims
- every feature mapped to status: exists / partial / planned

Outcome:
Trusted doc package.

Phase 3 — Controlled Backlog Generation

Priority: P1

Once the doc package passes QA:

- generate backend task list
- frontend task list
- DB/data task list
- integration task list
- security task list
- sales/marketing/support task list

Rules for backlog generation

Every generated task must include:

- exact output contract
- exact deliverable path or target
- acceptance check
- source references
- explicit feature/user-story linkage

Outcome:
Deterministic implementation backlog.

Phase 4 — Demo Readiness Pass

Priority: P1

Once the controlled backlog exists:

- create demo script
- create UI/demo journey map
- create sales narrative
- create support/onboarding flow
- create go/no-go checklist

Outcome:
Real demo package, not vague “prod-ready” claims.

Phase 5 — Production Readiness Pass

Priority: P2

Only after docs + QA + controlled backlog + demo prep:

- ops readiness verification

- secrets/process cleanup
- observability checks
- deployment validation
- test coverage/reliability review
- support and incident readiness

Outcome:
Production-readiness based on evidence, not optimism.

---

6. Immediate Next Actions

NOW-001 — Freeze stale architecture usage

- Treat `docs/ARCHITECTURE.md` as stale
- Do not use it for planning or QA

NOW-002 — Clean security hygiene issue

- Review and sanitize `infrastructure/README.md`
- Assume secret-like values may need rotation review

NOW-003 — Finalize canonical documentation baseline

- source-of-truth
- user stories
- feature baseline
- gaps/next steps
- QA checklist baseline

NOW-004 — Do not launch broad implementation waves yet

- no broad backend/frontend/db waves
- no mass task fan-out based on stale or mixed docs

NOW-005 — Generate only narrow, deterministic tasks

Until process reliability improves, every task should be:

- narrow,
- artifact-bound,
- source-referenced,
- QA-verifiable.

---

7. Success Condition

We are ready to resume the original plan when all of the following are true:

- authoritative vs stale docs are clearly defined
- canonical user stories exist
- canonical feature baseline exists
- canonical QA checklist exists
- documentation package passes QA
- implementation tasks can be generated from trusted sources only

---

8. Decision

PLOCK should proceed through a **documentation-truth-first** path, not a broad parallel execution wave.

This is not extra process for its own sake.
It is the minimum needed to ensure the next backlog is based on the real product rather than conflicting documentation or fabricated agent output.

Revision #3
Created 2026-03-14 12:03:43 UTC by John
Updated 2026-05-31 20:05:18 UTC by John