QA Checklist Baseline

Source: `~/ALAI/products/Plock/docs/demo-readiness/04-qa-checklist-baseline.md`

PLOCK — QA Checklist Baseline

Date: 2026-03-14
Purpose: Define the deterministic review checklist that must be applied to every PLOCK demo-readiness document before it is treated as usable input for planning, QA, demo prep, or backlog generation.

---

1. Why This Exists

PLOCK documentation work has already shown two failure modes:

1. documents can be generated from conflicting or stale sources, and
2. tasks can claim deliverables were created when the files do not actually exist.

This checklist prevents both.

A document is not considered approved because:

- an agent says it wrote it,
- a task says `done`, or
- the text sounds plausible.

A document is approved only if:

- the file exists,
- the path is correct,
- the content matches the requested deliverable,
- the claims are grounded in authoritative sources,
- contradictions have been checked,
- and the document passes the go/no-go review below.

2. Review Scope

Apply this checklist to every document under:

- `~/ALAI/products/Plock/docs/demo-readiness/`

At minimum, this includes:

- `00-canonical-source-of-truth.md`
- `01-user-stories-baseline.md`
- `02-feature-baseline.md`
- `03-gaps-and-next-steps.md`
- `04-qa-checklist-baseline.md`
- any later files added to the same package

---

3. Authority Order

When a demo-readiness document makes a claim, validate it using this precedence:

1. **Actual code and repo structure**
2. **Runtime configuration and DB migrations**
3. `BUILD-BLUEPRINT.md`
4. `README.md`
5. **Runbooks / security / ops docs**
6. **ADRs**
7. **PRD / GTM / Regulatory docs**
8. `docs/API-SPEC.md` **only if verified against code**
9. **Never use stale/conflicting docs as canonical truth**

Explicit stale-doc rule

`docs/ARCHITECTURE.md` is currently treated as **stale/conflicting** and must not be used as a canonical architecture source until rewritten or formally re-approved.

---

4. Required QA Outcome Labels

Each reviewed document must end in one of these states:

- **PASS** — safe to use as canonical planning input
- **PASS WITH NOTES** — usable, but with explicit caveats recorded
- **FAIL** — not safe to use; must be corrected before downstream work
- **BLOCKED** — cannot be reviewed properly because source evidence is missing or inaccessible

No silent approval is allowed.

---

5. Pre-Review Checks

Before reading content, verify:

5.1 File existence

- The file exists on disk.
- The file is readable.
- The filename matches the intended deliverable exactly.

5.2 Path correctness

- The file is located under the expected canonical path.
- If the task specified an exact path, the file is at that exact path.
- A similarly named file elsewhere does **not** count as pass.

5.3 Index discoverability

- `docs/demo-readiness/INDEX.md` links the file if it belongs in the canonical package.
- `docs/INDEX.md` points to the canonical package when appropriate.

5.4 Basic format sanity

- Markdown renders as a coherent document.
- Headings are structured.
- The document is not obviously truncated.
- Placeholder/TBD/filler text is either resolved or clearly marked as unresolved risk.

---

6. Artifact Verification Checklist

A document automatically fails if any of the following are true:

- claimed file does not exist
- wrong filename
- wrong directory
- empty or near-empty content
- content unrelated to requested deliverable
- generic narrative that does not match the task contract
- references to files, modules, or paths that do not exist without explicit “unverified” marking

Required artifact review questions

- Does the file exist exactly where promised?
- Is this the exact deliverable requested?
- Does the document content materially satisfy the task?
- Is it specific to PLOCK, not generic boilerplate?

---

7. Content Evidence Checklist

Every substantive claim should be supported by at least one of:

- repo path
- code module
- migration
- config file
- runbook/security/ADR reference
- product/business doc reference

Minimum evidence standard

For each major section, the reviewer should be able to answer:

- **What source supports this claim?**
- **Is that source authoritative?**
- **Does the claim overstate what the source proves?**

Preferred evidence style

Use explicit references such as:

- `backend/src/main/kotlin/...`
- `backend/src/main/resources/db/migration/...`
- `frontend/mfe-*`
- `docs/PRD.md`
- `docs/RLS-AUDIT.md`
- `README.md`
- `BUILD-BLUEPRINT.md`

---

8. Contradiction Check

The reviewer must actively search for contradictions between the reviewed document and stronger sources.

Mandatory contradiction checks

Verify the document does **not** incorrectly describe PLOCK as:

- Next.js instead of React/Vite MFEs
- Express instead of Ktor
- Prisma instead of Exposed + Flyway
- Supabase/Upstash as primary current stack without proof
- `organization_id` as the current tenant model where repo reality shows `warehouse_id`

Contradiction rule

If a claim conflicts with code or stronger docs, the document fails unless the conflict is explicitly called out as historical/stale context.

---

9. User Story and Feature Traceability Check

Every story or feature-oriented document must support traceability.

For user-story documents

Check that stories are:

- grounded in PRD and/or real code domains
- labeled clearly (`exists`, `partial`, `planned` or equivalent)
- not invented purely from marketing language
- not duplicated under different names without explanation

For feature documents

Check that each major feature maps back to at least one of:

- a route/service/module
- a migration/schema element
- a frontend MFE/module
- a product doc with explicit “planned” labeling

Required traceability rule

A feature that has no code evidence may still appear only if it is clearly labeled **planned** and anchored to an authoritative product/business doc.

---

10. Source-of-Truth Compliance Check

The reviewed document must align with `00-canonical-source-of-truth.md`.

Reviewer must verify:

- repo reality is treated as primary
- `docs/ARCHITECTURE.md` is not treated as canonical truth
- `docs/API-SPEC.md` is used carefully and only with verification caveat if needed
- agent-generated narrative is not cited as authoritative evidence

Automatic fail if the document:

- treats agent output as primary evidence
- uses stale architecture claims as current truth
- ignores stronger repo evidence

---

11. Security Hygiene Check

Every demo-readiness document that touches environments, infra, auth, secrets, integrations, or deployment must be reviewed for security hygiene.

Required checks

- No real secrets or secret-like material are copied into the doc.
- No environment variable values are presented in a way that looks reusable as production material.
- Any reference to `infrastructure/README.md` must respect the known hygiene issue around secret-like examples.
- If secret examples are mentioned, they must be described as hygiene risks, not copied forward as acceptable practice.
- Auth, token, encryption, and tenant-isolation claims must match real security docs/code where possible.

Automatic fail examples

- document repeats secret-looking example values as if acceptable
- document instructs users to commit secrets
- document misstates RLS/tenant isolation model
- document weakens existing security posture in prose without evidence

---

12. Canonical Path and Naming Check

The canonical package should remain predictable.

Reviewer should verify:

- numbering is consistent
- filenames are descriptive and stable
- index references are correct
- no duplicate “canonical” documents exist with competing names

Naming rule

If a document supersedes another document, the relationship must be explicit. Confusing parallel variants should fail review or be flagged for consolidation.

---

13. Acceptance Criteria by Document Type

13.1 Canonical source-of-truth docs

Must:

- identify authoritative vs stale sources
- define precedence clearly
- match repo reality
- explicitly quarantine stale architecture assumptions

13.2 User-story docs

Must:

- identify user roles clearly
- phrase stories in a usable operational way
- mark maturity/status
- avoid unsupported invention
- allow later mapping to features and backlog

13.3 Feature baseline docs

Must:

- group features coherently
- distinguish exists vs partial vs planned
- reference code/docs evidence
- capture key risks/contradictions

13.4 Gap / next-step docs

Must:

- separate current fact from recommendation

- prioritize gaps clearly
- define execution order
- avoid pretending that future work already exists

13.5 QA docs

Must:

- be deterministic
- define pass/fail rules
- include evidence expectations
- include final review workflow and go/no-go outcome

---

14. Reviewer Workflow

Use this exact workflow:

Step 1 — Confirm artifact

- file exists
- path is exact
- file is readable

Step 2 — Confirm task-fit

- compare document against requested deliverable
- reject generic filler or off-target content

Step 3 — Check evidence

- scan each major section for source grounding
- verify stronger-source precedence is respected

Step 4 — Check contradictions

- compare against repo reality and known stale-doc risks
- record any mismatch explicitly

Step 5 — Check traceability

- ensure user stories/features/gaps can be traced to evidence

Step 6 — Check security hygiene

- especially for env, infra, auth, tokens, integrations, deployment

Step 7 — Record verdict

Document one of:

- PASS
- PASS WITH NOTES
- FAIL
- BLOCKED

Step 8 — Record remediation

If not PASS, specify:

- exact issue
- exact file/section
- exact correction needed
- whether downstream work must pause

---

15. Review Record Template

Use this template for every reviewed document:

```
## QA Review Record
- Document:
- Path:
- Reviewer:
- Date:
- Verdict: PASS | PASS WITH NOTES | FAIL | BLOCKED

### Artifact Check
- Exists:
```

```
- Correct path:
- Correct filename:
- Indexed:

### Evidence Check
- Primary sources used:
- Any unsupported claims:

### Contradiction Check
- Conflicts found:
- Stale-doc contamination found:

### Traceability Check
- User-story traceability:
- Feature traceability:

### Security Hygiene Check
- Secret-like material copied: yes/no
- Security posture accurately described: yes/no

### Notes / Remediation
-
```

---

16. Go / No-Go Checklist

A demo-readiness document is **GO** only if all answers below are yes:

- Does the file exist at the exact canonical path?
- Does it match the requested deliverable?
- Is it grounded in authoritative sources?
- Does it avoid stale architecture assumptions?
- Are major claims evidence-backed?
- Are features/stories/gaps traceable?
- Are security-sensitive claims hygienic and accurate?
- Is the document specific enough to be used downstream?

If any answer is **no**, the outcome is **NO-GO**.

---

17. Package-Level Go / No-Go Rule

The full `docs/demo-readiness/` package is not ready for downstream backlog generation until:

- core baseline documents exist,
- each has passed review,
- no unresolved P0 contradiction remains,
- and no document relies on stale architecture as current truth.

Package-level blockers

Any of these are automatic package-level **NO-GO**:

- missing canonical file
- unresolved architecture contradiction
- unverified API assumptions treated as fact
- security hygiene issue copied into canonical docs
- major feature/story claims without evidence

---

18. Decision

This checklist is the minimum deterministic QA gate for PLOCK demo-readiness documentation.

No document in this package should be treated as approved planning input until it passes this checklist with an explicit recorded verdict.

Revision #3
Created 2026-03-14 12:03:44 UTC by John
Updated 2026-05-31 20:05:19 UTC by John