AI Code Review for Java (Spring Boot PRs) — Senior PR Review Pack (2026)

Checklist + Copy-Paste Prompts + PR Template to Catch Real Bugs

How to use (30 seconds): Run the checklist once before merging. Then use 2–3 prompts max on the riskiest files (service + data). Goal: prevent production bugs—not "perfect code".

Senior PR Checklist (Coach)

Correctness & edge cases (don't ship easy bugs)
□ Inputs/validation — invalid/empty/large inputs handled (DTO validation, bounds, enums).
Coach: If you can't state expected behavior for bad inputs, add validation or explicit errors.
□ Nullability — no hidden NPE paths; Optional used correctly; defaults explicit.
Coach: If "it should never be null", enforce it with validation/constructors.
□ Error handling — consistent exceptions → correct HTTP codes; no swallowed errors.
Coach: Inconsistent errors become on-call pain.
□ Business edge cases — duplicates, ordering, time zones, rounding, partial failures.
Coach: Add 1 test per risky edge case (even a thin integration test).

Data & transactions (where real bugs hide)
□ Transaction boundary — @Transactional (or equivalent) is correctly placed; rollback makes sense.
Coach: Keep consistency rules in the service layer.
□ DB efficiency — no N+1 queries; pagination/limits exist for lists.
Coach: Any "list" endpoint needs a default limit/pagination.
□ Idempotency & retries — retries won't create duplicates (keys, unique constraints, guards).
Coach: If clients can retry, your backend must be retry-safe.
□ Timeouts/resilience — external calls have timeouts; retries are bounded.
Coach: "No timeout" is a future incident.

Production + security (assume real users + attackers)
□ Observability — useful logs/metrics; correlation id; no secrets/PII; no noise.
Coach: Log decisions/failures, redact by default.
□ Performance foot-guns — loops over DB calls; large collections; memory growth; cache impact.
Coach: Remove obvious risks—don't optimize blindly.
□ AuthZ/AuthN — access checks correct (tenant scoping, ownership, method security).
Coach: "The UI won't call it" is not a security control.
□ Injection/leakage — sanitize inputs; secrets never logged; no data leakage.
Coach: Every log line can end up in a breach report.

AI Code Review for Java (Spring Boot PRs) — Senior PR Review Pack (2026)

Checklist + Copy-Paste Prompts + PR Template to Catch Real Bugs

How to use (30 seconds): Run the checklist once before merging. Then use 2–3 prompts max on the riskiest files (service + data). Goal: prevent production bugs—not "perfect code".

Copy-Paste Prompts (Use 2–3 max)

Paste once (context):
Context: Spring Boot app. Layered architecture (API/service/data).
Constraints: no breaking API changes; backwards compatible; production safety first.
Focus: correctness, transactions, performance, security, observability.
PR diff: {PASTE DIFF HERE}
Coach rule: run 2–3 prompts max. Start with data/transactions + security.

Core review

1. Senior review: top issues (P0–P2) with exact file refs + fixes.
2. Edge cases: missing failure modes + validations/tests.
3. API contract: codes, error model, backward compatibility.

Data / perf risks

4) Transactions: boundary + rollback + side effects.
5) DB efficiency: N+1, pagination, queries, indexes.
6) Concurrency/idempotency: duplicates, locking, retry-safe patterns.
7) Timeouts/retries: safe defaults + config placement.
8) Performance/memory: big collections, loops, hotspots + how to measure.

Security & leakage

9) AuthZ/tenant scoping: missing checks, cross-tenant leaks.
10) Input sanitization: injection risks, unsafe strings/headers.
11) Sensitive data: logs/exceptions/responses redaction.
12) LLM integration risks: prompt injection/data exfiltration guardrails.

PR Template

PR Description Template (Markdown)
Summary — What changed?
Why — What problem does it solve? (link ticket)
Changes — API / Service / Data / Config
Risks & Rollback — Low/Med/High + rollback steps
Observability — logs/metrics + dashboards/alerts
Testing Evidence — unit/integration/manual steps

Pre-merge checklist
□ Edge cases + validation
□ Transactions + DB efficiency
□ Timeouts/retries/idempotency
□ AuthZ + sensitive data