TEMPLATE

Programming Algorithm Review Template

A vendor-neutral template for evaluating whether an algorithmic approach fits the workload, failure mode, and operational constraints before implementation or deployment.

Programming Algorithm Review Template

Use this template to evaluate a programming solution in algorithmic terms before implementation or release. It is designed for technical teams that need to compare options for correctness, complexity, reliability, observability, and operational fit.

---

1) Problem statement

What problem are we solving? Describe the task in one sentence.

Example: Enrich incoming telemetry events with asset context before forwarding them to a detection engine.

Input(s): - - -

Expected output(s): - - -

Success criteria: - - -

---

2) Operational objective

Check the primary goal of the algorithmic choice.

- [ ] Correctness
- [ ] Low latency
- [ ] High throughput
- [ ] Low memory use
- [ ] Reliability / resilience
- [ ] Security / isolation
- [ ] Simplicity / maintainability
- [ ] Adaptability to changing workload

Primary objective: _____

Secondary objectives: _____

---

3) Workload characteristics

Document the conditions the algorithm must survive.

Factor	Notes
Data size	
Request rate	
Update frequency	
Read/write ratio	
Burst behavior	
Ordering requirements	
Time sensitivity	
Failure modes	

Dependency slowness / outages |
Growth expectation |

Observed or expected peak load: _____

Known constraints: _____

---

4) Candidate algorithm options

List the approaches you considered.

Option | Strategy | Benefits | Risks | Estimated complexity
A | | | |
B | | | |
C | | | |

Preferred option: _____

Why it was chosen: _____

---

5) Complexity and resource review

Assess the likely runtime and operational cost.

Time complexity: _____

Space complexity: _____

Hot path cost: _____

Memory pressure risk: _____

External calls / dependencies: _____

Expected bottlenecks: _____

---

6) Correctness checks

Use this section to verify that the algorithm behaves correctly in realistic conditions.

- [ ] Handles empty input
- [ ] Handles duplicates
- [ ] Handles malformed input
- [ ] Handles partial failure
- [ ] Handles overflow / underflow conditions
- [ ] Handles timeouts
- [ ] Preserves required ordering
- [ ] Produces deterministic output when required
- [ ] Avoids inconsistent state transitions
- [ ] Meets domain-specific invariants

Critical invariants: _____

Edge cases to test: - - -

---

7) Failure handling and recovery

What happens when dependencies slow down or fail?

_____

Fallback strategy:

- [ ] Fail closed
- [ ] Fail open
- [ ] Retry with backoff

- [ ] Return partial results
- [ ] Use cached data
- [ ] Degrade gracefully

Retry policy notes: _____

Rollback / kill switch available?

- [ ] Yes
- [ ] No

Notes: _____

---

8) Observability review

A production-ready algorithm should be measurable.

- [ ] Logs are sufficient for debugging
- [ ] Metrics capture latency, errors, and saturation
- [ ] Traces show dependency behavior
- [ ] Alerts exist for abnormal failure patterns
- [ ] Sensitive data is not exposed in logs or traces
- [ ] Performance baselines are defined

Signals to monitor: _____

Expected alert conditions: _____

---

9) Security and trust model

Review the algorithm for security-sensitive behavior.

- [ ] Input validation is explicit

- [ ] Authorization boundaries are respected
- [ ] Tenant boundaries are preserved
- [ ] Cache invalidation does not leak data
- [ ] Parsing is robust against malformed input
- [ ] Comparator / ordering logic is consistent
- [ ] No unsafe assumptions about input provenance

Security concerns: _____

Trust assumptions: _____

---

10) Production readiness gate

Before deployment, confirm the following:

- [ ] The algorithm matches the workload profile
- [ ] The implementation is bounded under expected load
- [ ] Edge cases have been tested
- [ ] Failure behavior is understood and acceptable
- [ ] Observability is in place
- [ ] Rollback path is documented
- [ ] Security implications have been reviewed
- [ ] The team can explain the trade-off to operators

Go / No-go decision:

- [ ] Go
- [ ] No-go

Decision notes: _____

---

11) Final review summary

What problem are we solving? _____

Which algorithmic approach fits best? _____

Main trade-off accepted: _____

Residual risk: _____

Owner: _____

Review date: _____

Quick reminder

A program may be syntactically correct and still be operationally wrong. Use this template to verify that the underlying algorithm is appropriate for the real workload, failure mode, and risk profile before you deploy it.