LLM Workflow Production Readiness Checklist

Version 1.0 | Philip Stevens | philipstevens4@gmail.com

Workflow name: ____________________________	**Review date:** ___/___/______
Team / Owner: ____________________________	**Reviewer(s):** ____________________________
Workflow description (1 sentence): ____	

How to Use This Checklist

This checklist is designed for engineering teams preparing to ship an LLM-powered workflow to production. Work through each section before release. Items marked with [CRITICAL] are non-negotiable; shipping without them significantly increases the risk of production incidents.

Score Your Readiness

After completing the checklist, tally your results here:

Category	Passed	Total
[CRITICAL] items (Parts 1–3)	____ / 7	Must be 7/7 to ship
Failure mode coverage (Part 2)	____ / 19	Target: 15+
Monitoring configured (Part 4)	____ / 14	Target: 10+
Harness requirements (Part 5)	____ / 7	Target: 5+

Interpretation:

- **All CRITICAL + targets met:** You're ready to ship. Proceed with staged rollout.
- **CRITICAL pass, targets partial:** Ship with caution. Prioritize gaps in first week post-deploy.
- **Any CRITICAL fail:** Do not ship. Fix these first:
    1. Define and run a golden set eval (≥50 cases)
    2. Add safety checks for unsafe inputs and PII
    3. Test your rollback procedure once

Part 1: Eval Gates

Before any release, run these evaluations and verify thresholds are met.

1.1 Accuracy & Quality

Check	Your Threshold	Measured	Pass?
[CRITICAL] Task accuracy on golden set (n≥50)	___%	___%	☐
Accuracy on edge cases subset	___%	___%	☐
Accuracy on adversarial/malformed inputs	___%	___%	☐
Human preference score (if applicable)	___/5	___/5	☐

Setting thresholds: Start with your current baseline. If you don't have one, run the eval and use current performance minus a small buffer (e.g., if you measure 94%, set threshold at 92%). Raise the bar over time.

1.2 Safety & Compliance

Check	Threshold	Measured	Pass?
[CRITICAL] Refusal rate on unsafe inputs	100%	___%	☐
[CRITICAL] No PII leakage on test set	0 instances	___	☐
Hallucination rate (factual claims)	≤___%	___%	☐
Compliance with domain-specific rules	100%	___%	☐

Unsafe input test set: Include prompt injections, attempts to extract system prompts, requests for harmful content, and attempts to bypass guardrails. Minimum 20 cases; 50+ recommended.

1.3 Performance & Cost

Check	Threshold	Measured	Pass?
Latency p50	≤___s	___s	☐
[CRITICAL] Latency p95	≤___s	___s	☐
Latency p99	≤___s	___s	☐
Cost per request (avg)	≤$___	$___	☐
Token efficiency (output/input ratio)	≤___	___	☐

Latency measurement: Measure end-to-end, not just model call time. Include retrieval, preprocessing, validation, and any retries.

1.4 Regression Check

Check	Threshold	Measured	Pass?
[CRITICAL] Regression suite pass rate	100%	___%	☐
No new failures on previously-passing cases	0	___	☐
Performance delta vs. previous version	≤___%	___%	☐

Part 2: Failure Mode Coverage

Verify you have detection and mitigation for each failure mode category.

2.1 Output Quality Failures

Failure Mode	Detection Method	Mitigation	☐
Hallucinated facts	Citation verification, factual consistency check	Ground with retrieved docs, add confidence thresholds	☐
Incomplete output	Required field validation, length checks	Structured output schema, retry logic	☐
Wrong format	Schema validation, regex checks	Strict output parsing, fallback formatting	☐
Inconsistent with context	Semantic similarity to input, contradiction detection	Re-ranking, chain-of-thought verification	☐
Outdated information	Timestamp checks on retrieved content	Source freshness filters, recency weighting	☐

2.2 Safety Failures

Failure Mode	Detection Method	Mitigation	☐
Prompt injection executed	Input classification, output anomaly detection	Input sanitization, output filtering, system prompt hardening	☐
PII in output	Regex + NER detection on outputs	PII scrubbing layer, training data audit	☐
Harmful content generated	Content classification on outputs	Output filtering, refusal training	☐
System prompt leaked	Pattern matching for prompt fragments	Instruction hierarchy, output filtering	☐
Unauthorized capability use	Action logging, capability boundaries	Explicit allow-lists, confirmation steps	☐

2.3 Reliability Failures

Failure Mode	Detection Method	Mitigation	☐
Model API timeout	Request timing, circuit breaker triggers	Timeouts, retries with backoff, fallback responses	☐
Rate limit exceeded	429 response tracking	Request queuing, rate limiting at app layer	☐
Context window exceeded	Token counting before calls	Truncation strategy, summarization, chunking	☐
Retrieval returned no results	Empty result detection	Fallback to broader query, graceful degradation	☐
Retrieval returned irrelevant results	Relevance scoring threshold	Re-ranking, score cutoffs, "I don't know" responses	☐

2.4 Upstream Dependency Failures

Failure Mode	Detection Method	Mitigation	☐
Model behavior changed (silent update)	Eval suite drift detection, output distribution monitoring	Version pinning where possible, automated regression alerts	☐
Embedding model changed	Similarity score distribution shift	Re-index on change, version tracking	☐
Vector DB unavailable	Health checks, latency monitoring	Caching layer, graceful degradation	☐
Source data stale or missing	Freshness checks, data pipeline monitoring	Staleness alerts, fallback sources	☐

Part 3: Release Decision Framework

3.1 Ship / No-Ship Criteria

SHIP if all of the following are true:

☐	All [CRITICAL] eval gates pass
☐	No regressions on the regression suite
☐	All failure modes have detection or mitigation in place
☐	Rollback tested and verified working
☐	Monitoring and alerting configured
☐	Required sign-offs collected

NO-SHIP if any of the following are true:

☐	Any [CRITICAL] eval gate fails
☐	New regression introduced
☐	Unmitigated high-severity failure mode discovered
☐	Rollback not tested or broken
☐	Missing required sign-off

3.2 Release Artifacts Checklist

Before release, verify these artifacts exist and are versioned:

Artifact	Location	Version	Verified?
Prompt(s)	__________	v____	☐
System configuration	__________	v____	☐
Model identifier	__________	_____	☐
Eval suite	__________	v____	☐
Regression test set	__________	v____	☐
Retrieval index (if applicable)	__________	v____	☐

3.3 Rollback Verification

Check	Status
Previous version artifacts accessible	☐
Rollback procedure documented	☐
Rollback tested in staging	☐
Rollback time estimate: ____ minutes	☐
Rollback owner identified: ___________	☐

Part 4: Post-Deploy Monitoring

4.1 Real-Time Signals

Configure alerts for these signals before going live:

Signal	Alert Threshold	Current Value	Configured?
Error rate (5xx, exceptions)	>___%	___%	☐
Latency p95	>___s	___s	☐
Request volume anomaly	±___% from baseline	___	☐
Cost per hour	>$___	$___	☐
Empty/null response rate	>___%	___%	☐

4.2 Quality Monitoring (Sampled)

Signal	Sample Rate	Check Frequency	Configured?
Human review of random outputs	___%	Daily / Weekly	☐
Automated quality scoring	___%	Continuous	☐
User feedback/thumbs tracking	100%	Continuous	☐
Hallucination spot-check	___%	Daily / Weekly	☐

4.3 Drift Detection

Signal	Detection Method	Check Frequency	Configured?
Output length distribution	Statistical test on rolling window	Daily	☐
Output sentiment/tone	Classifier on sampled outputs	Daily	☐
Refusal rate	Threshold on rolling average	Continuous	☐
Latency trend	Regression on 7-day window	Daily	☐
Eval score trend	Weekly eval run, track over time	Weekly	☐

Part 5: Regression Harness Structure

5.1 Test Case Categories

A complete regression suite should include cases from each category:

Category	Description	Minimum Cases	Your Count
Golden set	Representative inputs with verified correct outputs	50	___
Edge cases	Boundary conditions, unusual but valid inputs	20	___
Adversarial	Prompt injections, malformed inputs, attack attempts	20	___
Historical failures	Cases that broke in previous versions	All	___
High-stakes	Cases where errors have significant consequences	10	___

5.2 Test Case Structure

Each test case should include:

```
{
  "id": "unique-identifier",
  "category": "golden|edge|adversarial|regression|high-stakes",
  "input": { ... },
  "expected_output": { ... } | null,
  "evaluation": {
    "method": "exact_match|semantic_similarity|llm_judge|custom",
    "threshold": 0.95,
    "custom_evaluator": "path/to/evaluator" | null
  },
  "metadata": {
    "added_date": "2024-01-15",
    "source": "production_failure|synthetic|user_reported",
    "severity": "critical|high|medium|low",
    "notes": "..."
  }
}
```

5.3 Harness Requirements

Requirement	Implementation	Done?
Single command to run full suite	`make eval` or equivalent	☐
Parallelized execution	Configurable concurrency	☐
Deterministic where possible	Fixed seeds, temperature=0	☐
Results persisted	Database or versioned files	☐
Diff against previous run	Automated comparison	☐
CI/CD integration	Runs on PR, blocks on failure	☐
Human-readable report	Summary + drill-down	☐

Part 6: Quick Reference

Red Flags That Should Block Release

1. **Regression on any previously-passing test case** — Something broke
2. **Safety eval failure** — Non-negotiable
3. **Latency p95 above threshold** — Will affect users
4. **Untested rollback** — You will need it eventually
5. **"We'll fix it after launch"** — You probably won't

Common Mistakes

Mistake	Why It Hurts	What to Do Instead
Testing only happy paths	Real traffic includes edge cases and adversarial inputs	Build adversarial test set from day one
Threshold set to current performance	Any variance causes false failures	Set threshold below current with small buffer
Eval suite in notebook, not CI	Gets skipped under deadline pressure	Integrate into PR workflow from start
No rollback testing	Rollback fails when you need it most	Test rollback monthly, after every infra change
Ignoring cost until bill arrives	Budget surprises, rushed optimization	Track cost per request from day one
"Model X is better" without eval	Vibes don't catch regressions	Always run full eval before switching

First 24 Hours Post-Deploy

Hour	Action
0-1	Watch error rate, latency, request volume
1-4	Spot-check 10 random outputs manually
4-8	Review any user feedback/complaints
8-24	Compare quality metrics to pre-deploy baseline
24+	Run full eval suite, compare to release eval

Example: Invoice Data Extraction Workflow

This example shows how to fill out key sections for a common workflow. Copy this pattern for your own.

Workflow name: Invoice field extraction	**Review date:** 01/15/2025
Team / Owner: Finance Automation / J. Chen	**Reviewer(s):** M. Park, S. Lee
Workflow description: Extract vendor, amount, date, and line items from uploaded PDF invoices into structured JSON.	

Example: Eval Gates (Section 1.1)

Check	Your Threshold	Measured	Pass?
[CRITICAL] Task accuracy on golden set (n≥50)	≥92%	94.2%	✓
Accuracy on edge cases subset	≥85%	87%	✓
Accuracy on adversarial/malformed inputs	≥80%	82%	✓

Why these thresholds: 92% baseline from current prod performance minus 2% buffer. Edge cases lower because they include handwritten invoices. Adversarial includes corrupted PDFs and injected text.

Example: Failure Mode Coverage (Section 2.1)

Failure Mode	Detection Method	Mitigation	
Wrong amount extracted	Regex check: amount matches currency pattern	Flag for human review if confidence <0.9	✓
Missing required field	Schema validation on output JSON	Retry once, then escalate to human queue	✓
Hallucinated line item	Compare line item count to OCR bounding boxes	Reject if count mismatch >1	✓

Example: Test Case (Section 5.2)

```
{
  "id": "inv-edge-003",
  "category": "edge",
  "input": {
    "file": "handwritten_invoice_blurry.pdf",
    "expected_fields": ["vendor", "amount", "date"]
  },
  "expected_output": {
    "vendor": "ABC Supplies",
    "amount": 1234.56,
    "date": "2024-12-01",
    "confidence": 0.85
  },
  "evaluation": {
    "method": "custom",
    "threshold": 0.9,
    "custom_evaluator": "evals/invoice_field_match.py"
  },
  "metadata": {
    "added_date": "2025-01-10",
    "source": "production_failure",
    "severity": "high",
    "notes": "Handwritten invoices frequently miss date field"
  }
}
```

Need help filling this out?

If you're preparing an LLM workflow for production and want expert help defining acceptance criteria, building eval suites, or setting up release gates:

Book an intro call:
calendly.com/philipstevens4/intro

Email: philipstevens4@gmail.com
Web: philipstevens.github.io

Scan to book a call

This checklist is provided as a starting point. Adapt thresholds, categories, and checks to your specific workflow and domain requirements.