AI-Driven Local Phishing Detection Agent

Runtime, Execution Model & Versioned Roadmap (CPU-only)

1. Purpose of This Document

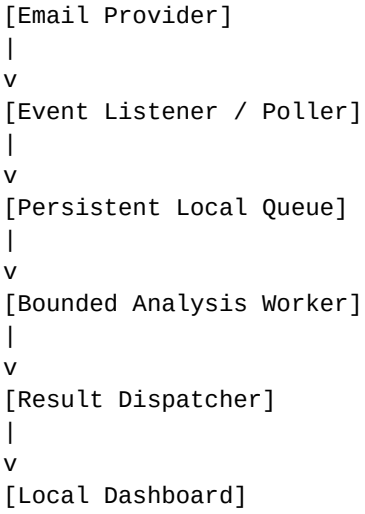
This document extends the core design by formally defining the runtime behavior, execution constraints, and performance-oriented design decisions for a local-only phishing detection agent. All techniques described assume CPU-only execution with strict resource bounds and privacy guarantees.

2. Execution Philosophy

1. Predictability over raw throughput
2. Bounded concurrency and controlled resource usage
3. Guaranteed processing of all emails (no drops)
4. Graceful degradation under load or network slowness
5. Explainable latency and user-visible progress

3. Runtime & Execution Architecture

The system follows a producer–consumer model where email arrival is decoupled from analysis. A persistent local queue ensures reliability, while bounded analysis workers prevent CPU or network exhaustion. All stages enforce timeouts and early-exit rules to maintain responsiveness.



4. Version 1 Runtime Model (v1)

1. Single analysis worker (serialized execution)
2. In-memory queue with disk-backed fallback
3. Hard timeouts for DNS, WHOIS, redirects, and ML inference
4. Early-exit logic when high-confidence phishing signals are detected
5. Model loaded once and reused across analyses
6. Basic TTL caching for DNS and domain metadata

Expected Behavior: If multiple emails arrive simultaneously, they are queued and analyzed one at a time. Alerts may be delayed by seconds, but no email is missed or dropped.

5. Version 2 Runtime Model (v2)

1. Optional second worker dedicated to network-bound analysis
2. Improved queue state tracking (PENDING, ANALYZING, DONE, FAILED)
3. Adaptive early-exit thresholds based on signal confidence
4. Expanded caching with configurable TTLs
5. Incremental UI updates to reflect analysis progress

Expected Behavior: Moderate bursts (5–10 emails) are absorbed smoothly with predictable latency. System remains responsive under normal desktop workloads.

6. Version 3 Runtime Model (v3)

1. Configurable worker pool with strict upper bounds
2. Advanced prioritization (e.g., suspicious emails first)
3. Optional headless sandbox with execution budget limits
4. Graceful throttling under sustained load
5. Preparation for optional future hardware acceleration (not enabled)

7. Recommended Techniques for This Scenario

1. Producer–consumer queue pattern for reliability
2. Timeout-driven network analysis to avoid hangs
3. Worst-link elevation for risk determination
4. Negative evidence detection for realism
5. Controlled vocabulary for explanations
6. TTL-based local caching to improve speed without persistence

8. Final Professional Assessment

With the addition of a formal runtime and execution model, the system design now reflects production-grade thinking. The chosen techniques balance security effectiveness, user trust, and local system constraints while preserving extensibility for future enhancements.