Product Requirements Document (PRD)

Sovereign AI Infrastructure: Bicameral Validator Ladder

Product Name: Sovereign AI Infrastructure
Version: 1.0
Date: February 5, 2026
Status: Draft for Review
Owner: Product Lead / Technical Lead
Stakeholders: Engineering Team, Operations, End Users, Management

---

Document Control

Version	Date	Author	Changes
1.0	2026-02-05	Product Team	Initial draft

Approval Sign-off:
- [ ] Product Lead
- [ ] Technical Lead
- [ ] Engineering Manager
- [ ] Security Architect
- [ ] Operations Lead

Executive Summary

The **Sovereign AI Infrastructure** is a local, multi-model AI orchestration system designed to deliver **enterprise-grade, validated AI outputs** while operating entirely on-premise with constrained hardware. The system addresses the critical challenge of deploying sophisticated AI capabilities within organizations that require **complete data sovereignty, transparent decision-making, and quality governance**.

The Problem

Current AI deployment options force organizations into an uncomfortable trade-off:
- **Cloud APIs** (OpenAI, Anthropic): Powerful but compromise data privacy, require ongoing costs, lack transparency
- **Single local models**: Preserve privacy but lack specialization, quality validation, and explainability

Organizations with **strict data sovereignty requirements** (healthcare, finance, legal, government, defense) cannot use cloud AI, yet single-model local solutions produce unreliable outputs without governance mechanisms.

The Solution

A **hardware-constrained, heterogeneous compute architecture** that orchestrates multiple specialist AI models with built-in validation, transparent decision trails, and multimodal capabilities—all running on a single workstation.

Core Innovation: Separate creative generation (GPU-resident Worker models) from critical validation (CPU-resident Validator and Router), achieving "sovereign-grade" quality through a **bicameral architecture** that mimics biological cognition (creative hemisphere + critical hemisphere).

Key Benefits

1. **Complete Data Sovereignty**: All computation local; no external dependencies
2. **Quality Governance**: Multi-layer validation ensures output reliability and safety
3. **Transparency & Auditability**: All decisions logged with human-readable reasoning
4. **Resource Efficiency**: Maximizes capability from constrained hardware (16GB VRAM)
5. **Cognitive Specialization**: Specialist models for coding, reasoning, creative work
6. **Multimodal Support**: Text, documents (OCR), images (vision), with provenance tracking

Target Users

- **Primary**: Organizations with strict data sovereignty requirements (healthcare, legal, finance, defense)
- **Secondary**: Technical teams requiring transparent, validated AI for high-stakes work
- **Tertiary**: Individual power users, researchers, privacy advocates

---

1. Product Vision & Objectives

1.1 Vision Statement

> "Empower organizations to deploy **sovereign-grade AI** that is local, transparent, validated, and trustworthy—without compromising capability for privacy."

1.2 Product Objectives

Primary Objectives

1. **Sovereignty**: Deliver 100% local AI inference with zero external dependencies
2. **Quality**: Achieve validated, reliable outputs through built-in governance layers
3. **Transparency**: Provide auditable decision trails for all AI actions
4. **Efficiency**: Maximize capability within hardware constraints (16GB VRAM, 128GB RAM)
5. **Specialization**: Enable cognitive specialists to outperform general-purpose models

Secondary Objectives

1. **Multimodal**: Support text, documents, and images with grounding
2. **Scalability**: Architecture extensible to multi-GPU, distributed setups (future)
3. **Maintainability**: Understandable, debuggable, operationally sustainable
4. **Usability**: Accessible to domain experts, not just ML engineers

1.3 Success Criteria (High-Level)

The product is considered successful if:
- [ ] It enables organizations to **replace cloud AI** for sensitive workloads

- [ ] Users **trust the outputs** more than single-model alternatives
- [ ] **Regulatory compliance** is achieved (data sovereignty, auditability)
- [ ] System is **operationally stable** (99%+ uptime) in production
- [ ] **Resource constraints** are respected (no hardware upgrades required)

---

2. Target Users & Use Cases

2.1 Target User Personas

Persona 1: "The Compliance Officer" (Primary)

- **Role**: CISO, Compliance Manager, Legal Counsel
- **Organization**: Healthcare provider, financial institution, law firm
- **Pain Points**:
- Cannot use cloud AI due to data privacy regulations (HIPAA, GDPR, SOC 2)
- Needs AI capabilities for document analysis, contract review, medical coding
- Requires audit trails for regulatory compliance
- Distrusts "black box" AI decisions
- **Goals**:
- Deploy AI while maintaining full data control
- Demonstrate compliance to auditors (transparent, logged decisions)
- Ensure AI outputs are validated and reliable
- **Success Metrics**: Passes compliance audits, data never leaves premises, all decisions auditable

Persona 2: "The Technical Lead" (Primary)

- **Role**: Engineering Manager, Staff Engineer, Solutions Architect
- **Organization**: Software company, research lab, tech-forward enterprise
- **Pain Points**:
- Single general-purpose models produce mediocre code/reasoning
- No built-in validation catches hallucinations or errors
- Iterative refinement is manual and time-consuming
- Lacks transparency into model decisions
- **Goals**:
- Leverage specialist models for architecture, implementation, validation
- Automated quality checks (validation as code)
- Transparent decision trails for debugging
- High-quality outputs that reduce manual review
- **Success Metrics**: Fewer bugs in AI-generated code, faster development cycles, trusted AI collaboration

Persona 3: "The Researcher" (Secondary)

- **Role**: PhD student, academic researcher, data scientist
- **Organization**: University, research institute, R&D lab
- **Pain Points**:
- Budget constraints (cloud API costs)
- Need full control over model behavior for experiments
- Requires reproducibility and transparency

- Multimodal analysis (text + images + documents)
- **Goals**:
- Cost-effective local inference
- Customizable routing and validation logic
- Transparent, reproducible experiments
- Multimodal capabilities for research
- **Success Metrics**: Zero ongoing API costs, experiments reproducible, full transparency into decisions

Persona 4: “The Privacy Advocate” (Tertiary)

- **Role**: Individual user, privacy-conscious professional
- **Organization**: Self-employed, journalist, activist
- **Pain Points**:
- Cloud AI lacks privacy (data harvested for training)
- Distrusts proprietary models (alignment, bias, censorship)
- Wants control over AI capabilities
- **Goals**:
- Complete data privacy (local-only)
- Open-weight models (inspectable, auditable)
- No vendor lock-in
- **Success Metrics**: Data never transmitted externally, full control over models, no censorship

2.2 Core Use Cases

Use Case 1: Validated Code Generation for High-Stakes Systems

Actor: Technical Lead
Scenario: Generate and validate production code for safety-critical systems (medical devices, financial infrastructure)

Flow:
1. User submits code generation request (e.g., “Implement payment processing module with fraud detection”)
2. Router classifies as “high-stakes coding” → selects Qwen Coder (architecture) + Nemotron (implementation) + Granite-H (validator)
3. Qwen Coder generates architecture design (classes, interfaces, data flows)
4. Validator reviews architecture for logical consistency, missing edge cases, security flaws → PASS or FAIL with corrections
5. Nemotron generates implementation (optimized, performant code)
6. Validator reviews implementation for correctness, performance issues, security vulnerabilities → PASS or FAIL
7. Final output: Validated code + validation report with all checks and reasoning

Success Criteria:
- [ ] Generated code passes validation without critical errors
- [ ] Validation catches real issues (no false negatives)
- [ ] Validation does not over-reject correct code (false positives <5%)
- [ ] Output includes transparent audit trail (why this design, why these checks)
- [ ] User trusts output enough to deploy with minimal manual review

---

Use Case 2: Compliant Document Analysis (Healthcare/Legal)

Actor: Compliance Officer
Scenario: Analyze scanned medical records or legal contracts, extract key information, ensure compliance

Flow:
1. User uploads scanned document (PDF, image)
2. Router detects document input → routes to OCR pipeline
3. OCR extracts text with provenance (page numbers, bounding boxes)
4. Router classifies task → GPT-OSS (reasoning/extraction) + Granite-H (validation)
5. GPT-OSS extracts: patient info, diagnoses, medications, key dates (legal: parties, clauses, obligations)
6. Validator checks: Are all claims grounded in OCR text? Are citations correct? Any hallucinations?
7. Validator flags ungrounded claims or uncertain extractions
8. Final output: Structured data + grounding citations + confidence scores + audit trail

Success Criteria:
- [ ] OCR extraction ≥90% accurate
- [ ] All extracted claims are grounded in source document (traceable)
- [ ] Validator catches hallucinations (no fabricated information)
- [ ] Audit trail shows: OCR confidence, extraction reasoning, validation checks
- [ ] Compliance officers can demonstrate to auditors that no data left premises

---

Use Case 3: Multimodal Research Analysis (Image + Text)

Actor: Researcher
Scenario: Analyze scientific papers with figures/diagrams, generate research summary

Flow:
1. User provides research paper (PDF with images/charts)
2. Router detects multimodal input → routes to OCR (text) + Vision (images)
3. OCR extracts paper text; Vision generates captions for figures/charts
4. Router → GPT-OSS (reasoning/synthesis)
5. GPT-OSS generates research summary, citing text paragraphs and figure descriptions
6. Validator checks: Are text citations correct? Are figure descriptions aligned with captions?
7. Final output: Research summary + citations (text + figures) + audit trail

Success Criteria:
- [ ] Multimodal inputs processed correctly (text + images)
- [ ] Summary accurately reflects paper content
- [ ] All claims grounded in source (text or image)
- [ ] Provenance tracked (which page, which figure)
- [ ] Researcher can verify every claim against source

Use Case 4: Creative Content Generation with Brand Safety

Actor: Marketing team (enterprise)
Scenario: Generate creative marketing copy that adheres to brand guidelines and safety policies

Flow:
1. User submits creative brief (e.g., “Write product announcement for new AI tool, professional tone, no jargon”)
2. Router classifies as “creative, medium stakes” → MythoMax (creative) + Granite-H (validator)
3. MythoMax generates draft copy
4. Validator checks: Tone adherence, brand guideline compliance, safety (no harmful content), clarity
5. If violations → FAIL with corrections, MythoMax retries
6. Final output: Approved copy + validation checks + reasoning

Success Criteria:
- [ ] Creative output is high-quality, engaging
- [ ] All brand guidelines respected (no violations)
- [ ] No unsafe/harmful content
- [ ] Validator catches policy violations before output reaches user
- [ ] Faster than manual review cycles

Use Case 5: Transparent Routing for Ambiguous Queries

Actor: Technical Lead
Scenario: User submits ambiguous query (could be coding, reasoning, or both)

Flow:
1. User submits query (e.g., “How do I optimize database queries for real-time analytics?”)
2. Router analyzes query → uncertain (could be architecture OR implementation)
3. Router checks confidence score → LOW → triggers fallback: embedding-based similarity search
4. Embedding retrieval suggests similar past queries → identified as “architecture design”
5. Router selects Qwen Coder (architecture specialist)
6. If still uncertain → Router logs “uncertainty note” in audit trail, defaults to safer option (GPT-OSS general reasoning)
7. Output includes explanation: “Routed to Qwen Coder (architecture) based on similarity to past query X; router confidence: 72%”

Success Criteria:
- [ ] Router handles ambiguous queries gracefully (no failures)
- [ ] Confidence scoring accurate (low confidence correlates with actual ambiguity)
- [ ] Fallback mechanisms work (embedding similarity, safe defaults)
- [ ] Uncertainty is transparent to user (logged and explained)
- [ ] User can override routing decision if desired

2.3 Out-of-Scope Use Cases (for v1.0)

The following are **explicitly out of scope** for the initial release:

1. **Real-time inference (<1 second)**: This system prioritizes quality over speed; not suitable for chatbots requiring instant responses
2. **Multi-user concurrent access**: Single-user or small team; not designed for large-scale concurrent load
3. **Fine-tuning models**: Uses pre-trained models; fine-tuning capability not included

4. **Distributed multi-GPU setups**: Single workstation only; horizontal scaling deferred to future versions
5. **Cloud deployment**: Designed for on-premise; cloud-native deployment not a v1.0 goal
6. **GUI/Web interface**: CLI or API only; graphical UI deferred to future
7. **Voice/audio input**: Text, document, image only; audio/video not supported
8. **Streaming outputs**: Batch generation only; token-by-token streaming not prioritized

---

3. Functional Requirements

3.1 Core System Capabilities

FR-1: Multi-Model Orchestration

Priority: P0 (Critical)
Description: The system shall orchestrate multiple specialist AI models, routing requests to the appropriate model based on task characteristics.

Acceptance Criteria:
- [ ] System supports minimum 4 specialist Worker models (coding, reasoning, creative, implementation)
- [ ] System supports 1 Router model (intent classification)
- [ ] System supports 1 Validator model (quality governance)
- [ ] Only one Worker model loaded in GPU VRAM at a time
- [ ] Router and Validator are CPU-resident (always loaded)
- [ ] Model swapping completes in ≤5 seconds

FR-2: Intelligent Request Routing

Priority: P0 (Critical)
Description: The system shall analyze incoming requests and route them to the most appropriate specialist model based on domain, stakes, and task characteristics.

Acceptance Criteria:
- [ ] Router classifies requests into domains: coding (architecture), coding (implementation), reasoning, creative, documentation
- [ ] Router assesses stakes: low, medium, high (based on complexity, risk indicators)
- [ ] Router selects appropriate Worker model with ≥85% accuracy
- [ ] Router outputs confidence score (0-100%)
- [ ] Router handles ambiguous requests via fallback mechanisms (embedding similarity, safe defaults)
- [ ] Router logs every decision with human-readable reasoning
- [ ] Router decisions are deterministic (same input → same output)

FR-3: Line-by-Line Validation (Proof-Checking)

Priority: P0 (Critical)
Description: For high-stakes tasks, the system shall validate Worker outputs incrementally (block-by-block or line-by-line) to catch errors early, mimicking mathematical proof verification.

Acceptance Criteria:
- [ ] Validation can be configured at multiple granularities: per-line, per-block (5-10 lines), per-function, per-stage
- [ ] Validator checks each block against: project state, logical consistency, hallucination indicators, syntax errors (code), policy compliance
- [ ] Validator outputs: `[PASS]` or `[FAIL: specific reason]`
- [ ] Failed blocks trigger retry: Validator writes correction directive to scratchpad, Worker retries
- [ ] Maximum 3 retries per block before escalation or user intervention
- [ ] Validation does not require GPU model swaps (CPU-resident Validator)
- [ ] Validation latency: ≤5 seconds per block

FR-4: Markdown Memory Ledger

Priority: P0 (Critical)
Description: The system shall maintain a transparent, human-readable memory system using Markdown files as the shared “brain” across models.

Acceptance Criteria:
- [ ] System maintains 3 memory files:
- `project_state.md`: Immutable facts, global objectives, proven constraints
- `scratchpad.md`: Active reasoning, pending steps, validator feedback, retry attempts
- `knowledge_graph.md`: Learned patterns, model-specific behaviors, domain knowledge
- [ ] All models (Router, Worker, Validator) read from and write to memory files
- [ ] Memory files are version-controlled (Git-compatible)
- [ ] Memory files are human-readable (domain experts can inspect)
- [ ] Memory updates are atomic (no partial writes, corruption)
- [ ] Memory system supports sessions: save state, resume later

---

FR-5: Bicameral GPU/CPU Architecture

Priority: P0 (Critical)
Description: The system shall separate creative generation (GPU) from critical validation (CPU) to avoid model thrashing and enable zero-swap validation.

Acceptance Criteria:
- [ ] Worker models (generation) run on GPU (VRAM-resident)
- [ ] Router and Validator run on CPU (RAM-resident, always loaded)
- [ ] Worker can generate while Validator is checking previous block (parallel execution where possible)
- [ ] No GPU unload required for validation (Validator on CPU eliminates swap)
- [ ] CPU can sustain Validator at ≥3 tokens/second
- [ ] GPU and CPU can execute concurrently (no blocking)

FR-6: Warm Pool Strategy (Predictive Model Pre-loading)

Priority: P1 (High)
Description: The system shall pre-load likely-next models into RAM to minimize latency when switching specialists.

Acceptance Criteria:
- [ ] System maintains “warm pool” of models in system RAM (not VRAM)
- [ ] Warm pool size configurable (default: 2-3 models)
- [ ] Pre-loading logic predicts next model based on current domain, user history, task patterns
- [ ] Loading from RAM to VRAM: ≤3 seconds (vs. ≥10 seconds from NVMe)
- [ ] Eviction policy: LRU (Least Recently Used) or domain-based priority
- [ ] Warm pool adapts based on actual usage patterns (learn over time)

FR-7: Multimodal Input Processing

Priority: P1 (High)
Description: The system shall accept and process text, document images (OCR), and visual images (vision encoders), with full provenance tracking.

Sub-requirements:

FR-7a: Optical Character Recognition (OCR)
- [ ] System accepts scanned documents (PDF, JPG, PNG)
- [ ] OCR extracts text with ≥90% accuracy (standard printed documents)
- [ ] OCR output includes: extracted text, bounding boxes (if available), page numbers, confidence scores
- [ ] OCR failures/uncertainties are flagged explicitly (“unreadable region on page 3”)
- [ ] OCR provenance tracked: all extracted text tagged with source image/page reference

FR-7b: Vision Encoding (Image Understanding)
- [ ] System accepts image inputs (JPG, PNG)
- [ ] Vision encoder generates: dense caption, object list, layout description, visual features
- [ ] Vision output quality: accurate descriptions (subjectively assessed)
- [ ] Vision failures/uncertainties flagged (“image unclear, low confidence”)
- [ ] Vision provenance tracked: all descriptions tagged with source image

FR-7c: Embedding Models (Semantic Retrieval)
- [ ] System embeds text queries into vector space
- [ ] Retrieval from indexed knowledge base (Markdown memory, documentation, past tasks)
- [ ] Top-k retrieval (k=5-10) completes in ≤2 seconds
- [ ] Retrieval used for: Router fallback (ambiguous queries), Worker context augmentation, Validator grounding checks
- [ ] Embeddings updated incrementally as memory grows

FR-7d: Multimodal Routing
- [ ] Router detects input modality (text, document, image, mixed)
- [ ] Router triggers appropriate pipelines (OCR, vision, embeddings) before Worker invocation
- [ ] Multimodal inputs processed sequentially: OCR/vision first → structured output → Worker generation

---

FR-8: Grounding and Provenance Tracking

Priority: P1 (High)
Description: For multimodal inputs (OCR, vision), the system shall ensure all claims are grounded in source material and track provenance.

Acceptance Criteria:
- [ ] All Worker outputs citing OCR text include: source page number, excerpt
- [ ] All Worker outputs citing image content include: source image filename, caption reference
- [ ] Validator checks grounding: "Is this claim supported by OCR text or vision output?"
- [ ] Ungrounded claims are flagged: `[WARNING: Ungrounded claim: "X" - no source found]`
- [ ] Grounding accuracy: ≥80% of claims correctly attributed to source
- [ ] Provenance logged in memory ledger (audit trail)

FR-9: Configurable Validation Policies

Priority: P1 (High)
Description: The system shall support stakes-based validation policies, allowing users to configure validation rigor based on task risk.

Acceptance Criteria:
- [ ] Three validation policies:
- **Low stakes**: Optional validation, single-pass generation
- **Medium stakes**: Mandatory end-of-stage validation
- **High stakes**: Block-by-block validation (proof-checking mode)
- [ ] Policies configurable per request (user can specify) or auto-selected by Router
- [ ] Policies include: granularity (line, block, stage), validator model, retry limits, grounding requirements
- [ ] Policy changes logged in audit trail

FR-10: Transparent Audit Trail

Priority: P0 (Critical)
Description: The system shall log all decisions, actions, and reasoning in a human-readable audit trail for compliance and debugging.

Acceptance Criteria:
- [ ] Audit trail includes:
- Routing decision (why this model was chosen)
- Validation results (pass/fail, reasoning)
- Tool invocations (OCR, vision, embeddings)
- Grounding checks (source citations)
- Retry attempts and corrections
- Uncertainty notes (low-confidence decisions)
- [ ] Audit trail stored in Markdown memory ledger
- [ ] Audit trail is immutable (append-only)
- [ ] Audit trail is timestamped (UTC)
- [ ] Audit trail is searchable (text search, grep-compatible)
- [ ] Audit trail can be exported (PDF, HTML) for compliance reports

---

FR-11: Failure Recovery and Resilience

Priority: P1 (High)
Description: The system shall handle failures gracefully (OOM, model swap errors, validation loops) without data loss or corruption.

Acceptance Criteria:
- [ ] Out-of-Memory (OOM) detection: System monitors VRAM usage, prevents OOM crashes
- [ ] OOM recovery: Fallback to more aggressively quantized model or smaller alternative
- [ ] Model swap failures: Retry with exponential backoff (3 attempts); log error if persistent
- [ ] Validation infinite loops: Maximum retry limit (3 per block); escalate to user or abort task
- [ ] Memory file corruption: Integrity checks on read; restore from backup if corrupted
- [ ] All failures logged with stack traces and context
- [ ] System state recoverable: Resume from last committed block

---

3.2 Specialist Model Requirements

FR-12: Coding Specialist (Architecture)

Model: Qwen Coder 32B or equivalent
Priority: P0

Capabilities:
- [ ] Deep multi-file codebase reasoning
- [ ] Architecture design and refactoring
- [ ] Explanation of complex code patterns
- [ ] Language support: Python, C++, Java, JavaScript/TypeScript, Go, Rust (minimum)

Performance:
- [ ] VRAM footprint: ≤18GB (Q4_K_M quantization)
- [ ] Inference speed: ≥20 tokens/second
- [ ] Context window: ≥4K tokens

FR-13: Coding Specialist (Implementation)

Model: Nemotron-3 Nano 30B or equivalent
Priority: P0

Capabilities:
- [ ] Performance-optimized code generation
- [ ] Practical, working implementations (minimal hallucination)
- [ ] Operational coding (scripts, utilities, pipelines)
- [ ] Language support: Same as FR-12

Performance:
- [ ] VRAM footprint: ≤16GB (Q4_K_M quantization)
- [ ] Inference speed: ≥25 tokens/second (optimized for Tesla A2)
- [ ] Context window: ≥4K tokens

FR-14: Reasoning Specialist

Model: GPT-OSS 20B or equivalent MoE
Priority: P0

Capabilities:
- [ ] General reasoning, planning, problem decomposition
- [ ] Instruction following with high fidelity
- [ ] Tool use and API calls (if applicable)
- [ ] Multi-step logical reasoning

Performance:
- [ ] VRAM footprint: ≤12GB (Q4_K_M, MoE advantage)
- [ ] Inference speed: ≥30 tokens/second (due to sparse activation)
- [ ] Context window: ≥4K tokens

FR-15: Creative Specialist

Model: MythoMax-L2-13B or equivalent
Priority: P1

Capabilities:
- [ ] Creative writing, narrative, storytelling
- [ ] Tone and style variation
- [ ] Engaging, non-monotonous prose

Performance:
- [ ] VRAM footprint: ≤9GB (Q5_K_M for higher quality)
- [ ] Inference speed: ≥30 tokens/second
- [ ] Context window: ≥4K tokens

FR-16: Validator Specialist

Model: Granite-H-Small (MoE, 9B active) or equivalent
Priority: P0

Capabilities:
- [ ] Strict instruction adherence
- [ ] Error detection: logical errors, hallucinations, policy violations, syntax errors
- [ ] Summarization and structure review
- [ ] Safety and compliance checking

Performance:
- [ ] Deployed on CPU (RAM-resident, always loaded)
- [ ] Inference speed: ≥3 tokens/second on CPU (acceptable for validation)
- [ ] RAM footprint: ≤20GB (Q4_K_M quantization)
- [ ] Output format: `[PASS]` or `[FAIL: reason]` (structured, parseable)

FR-17: Router Specialist

Model: Granite-Micro 3B or equivalent
Priority: P0

Capabilities:
- [ ] Intent classification (domain, stakes, task type)
- [ ] JSON-structured output (parseable routing decisions)
- [ ] Fast inference (classification latency ≤1 second)

Performance:
- [ ] Deployed on CPU (RAM-resident, always loaded)
- [ ] Inference speed: ≥10 tokens/second on CPU
- [ ] RAM footprint: ≤6GB (FP16 or Q8)
- [ ] Output format: JSON with keys: `domain`, `stakes`, `model`, `tools_required`, `confidence`

3.3 Integration & API Requirements

FR-18: API Interfaces

Priority: P1 (High)

Description: The system shall expose programmatic interfaces for integration with other tools.

Acceptance Criteria:
- [ ] REST API (HTTP endpoints):
- `POST /infer` - Submit task, receive output
- `GET /status` - Check system health, model status
- `GET /history` - Retrieve audit trail / past tasks
- `POST /feedback` - Submit user feedback on outputs
- [ ] API authentication: API key or token-based (for multi-user environments)
- [ ] API rate limiting: Configurable (default: 10 requests/minute per user)
- [ ] API documentation: OpenAPI 3.0 spec, Swagger UI
- [ ] Error responses: Structured JSON with error codes, messages

---

FR-19: Command-Line Interface (CLI)

Priority: P1 (High)

Description: The system shall provide a CLI for direct user interaction.

Acceptance Criteria:
- [ ] CLI commands:
- `sovereign infer <task>` - Submit task, stream output
- `sovereign status` - System health check
- `sovereign history` - View past tasks
- `sovereign logs` - View audit trail
- `sovereign config` - View/edit configuration
- [ ] CLI supports interactive mode (prompt-based)
- [ ] CLI supports batch mode (file input)
- [ ] CLI outputs: Plain text (default), JSON (optional), Markdown (optional)

FR-20: Configuration Management

Priority: P1 (High)

Description: The system shall support flexible configuration for models, policies, hardware settings.

Acceptance Criteria:
- [ ] Configuration file format: YAML or TOML (human-readable)
- [ ] Configuration includes:
- Model paths and quantization levels
- Hardware allocation (VRAM limits, RAM limits, CPU threads)
- Validation policies (stakes thresholds, granularity)
- Warm pool settings (size, eviction policy)
- Logging and monitoring settings
- [ ] Configuration validation on startup (catch errors early)
- [ ] Configuration hot-reload: Changes applied without full restart (where safe)
- [ ] Default configuration provided (works out-of-box for specified hardware)

4. Non-Functional Requirements

4.1 Performance Requirements

NFR-1: Latency

Priority: P0 (Critical)

Task Type	Target Latency	Acceptable Latency	Unacceptable
Router classification	≤1 second	≤2 seconds	>3 seconds
Simple generation (low stakes)	≤10 seconds	≤20 seconds	>30 seconds
Complex generation (medium stakes)	≤30 seconds	≤60 seconds	>120 seconds
High-stakes (block validation)	≤60 seconds	≤120 seconds	>180 seconds
Model swap (RAM → VRAM)	≤3 seconds	≤5 seconds	>10 seconds
OCR processing (per page)	≤5 seconds	≤10 seconds	>20 seconds
Vision encoding (per image)	≤3 seconds	≤5 seconds	>10 seconds
Embedding retrieval	≤1 second	≤2 seconds	>5 seconds

Measurement: Latency measured from user request to final output (end-to-end).

NFR-2: Throughput

Priority: P1 (High)

Metric	Target	Acceptable	Unacceptable
Tasks per hour (sequential)	≥30	≥20	<10
Worker inference speed	≥20 tokens/sec	≥15 tok/s	<10 tok/s
Validator inference speed (CPU)	≥3 tokens/sec	≥2 tok/s	<1 tok/s
Router inference speed (CPU)	≥10 tokens/sec	≥5 tok/s	<3 tok/s

Note: Sequential throughput (not concurrent); concurrent multi-user support is out of scope for v1.0.

NFR-3: Resource Utilization

Priority: P0 (Critical)

Hardware Constraints (must not exceed):
- **VRAM**: ≤16GB (Tesla A2 limit)
- **RAM**: ≤120GB of 128GB available (leave 8GB for OS/other processes)
- **NVMe Storage**: ≥500GB free (for model vault)
- **CPU**: ≤80% sustained utilization (leave headroom for OS, monitoring)

Resource Efficiency Targets:
- [ ] Worker model VRAM footprint: 12-18GB (depending on model size)
- [ ] Router + Validator RAM footprint: ≤26GB combined
- [ ] Warm pool RAM footprint: ≤60GB (2-3 models)
- [ ] NVMe read operations: Minimize (warm pool should reduce cold starts)

Monitoring:
- [ ] Real-time VRAM monitoring (prevent OOM)
- [ ] RAM monitoring (detect swap to disk)
- [ ] Disk I/O monitoring (track model loading frequency)
- [ ] CPU temperature monitoring (detect thermal throttling)

4.2 Accuracy & Quality Requirements

NFR-4: Router Accuracy

Priority: P0 (Critical)

Metric	Target	Acceptable	Unacceptable
Domain classification accuracy	≥90%	≥85%	<80%
Stakes assessment accuracy	≥85%	≥75%	<70%
Model selection accuracy	≥90%	≥85%	<80%
Confidence calibration (low confidence correlates with errors)	≥80%	≥70%	<60%

Measurement: Accuracy measured against manually-labeled test dataset (≥200 prompts).

NFR-5: Validator Accuracy

Priority: P0 (Critical)

Metric	Target	Acceptable	Unacceptable
False Positive rate (rejects good outputs)	<5%	<10%	≥15%
False Negative rate (misses real errors)	<3%	<5%	≥10%
True Positive rate (catches real errors)	≥95%	≥90%	<85%
True Negative rate (approves good outputs)	≥95%	≥90%	<85%

Measurement: Accuracy measured against manually-labeled test dataset (≥100 code blocks: 50 correct, 50 with known errors).

NFR-6: Multimodal Accuracy

Priority: P1 (High)

Metric	Target	Acceptable	Unacceptable
OCR text extraction accuracy	≥95%	≥90%	<85%
Vision caption accuracy (subjective)	≥80%	≥70%	<60%
Grounding accuracy (claims correctly attributed)	≥85%	≥80%	<70%
Provenance tracking accuracy (sources logged)	≥95%	≥90%	<85%

Measurement: OCR accuracy via character-level or word-level edit distance vs. ground truth. Vision and grounding accuracy via manual subjective assessment.

4.3 Reliability & Availability

NFR-7: System Uptime

Priority: P0 (Critical)

Target: ≥99% uptime in production (first 3 months)
Acceptable: ≥95% uptime
Unacceptable: <90% uptime

Downtime categories:
- **Planned maintenance**: Excluded from uptime calculation (must be scheduled, communicated)
- **Unplanned outages**: Included (crashes, OOM, hardware failures)

Measurement: Uptime = (Total time - Unplanned downtime) / Total time

---

NFR-8: Failure Recovery

Priority: P1 (High)

Recovery Time Objective (RTO): ≤5 minutes (time to restore service after failure)
Recovery Point Objective (RPO): ≤1 task (maximum data loss: in-progress task only; completed tasks must be preserved)

Failure scenarios:
- [ ] GPU crash: System restarts GPU, reloads last Worker model, resumes from last checkpoint
- [ ] OOM: System unloads Worker, loads smaller model or more aggressive quantization, retries task
- [ ] Memory file corruption: System detects corruption, restores from backup (Git history), logs incident
- [ ] Model swap failure: System retries swap (3 attempts), falls back to cached model, logs error

---

NFR-9: Data Integrity

Priority: P0 (Critical)

Requirements:
- [ ] Memory files (Markdown ledger) must never be partially written (atomic writes)
- [ ] Audit trail must be append-only (no deletions, no silent modifications)
- [ ] Git version control for memory files (every task commits state)
- [ ] Checksums for model files (detect corruption, trigger re-download)
- [ ] Backup strategy: Daily backups of memory files, weekly backups of model vault

---

4.4 Security & Privacy

NFR-10: Data Sovereignty

Priority: P0 (Critical)

Requirements:
- [ ] **Zero external dependencies**: No API calls to external services (no cloud models, no telemetry)
- [ ] **Local-only computation**: All inference, storage, and processing on-premise
- [ ] **No internet requirement**: System operates fully offline (except for initial model downloads)
- [ ] **Data never transmitted**: User data, prompts, outputs, memory files never leave local machine
- [ ] **Compliance**: System design supports HIPAA, GDPR, SOC 2 compliance (by ensuring data sovereignty)

Verification: Network monitoring confirms zero external traffic during operation (excluding model downloads).

NFR-11: Access Control

Priority: P1 (High)

Requirements (for multi-user environments):
- [ ] API authentication: Token-based (API keys) or OAuth
- [ ] Role-based access control (RBAC): Admin, User, Read-only
- [ ] Audit logging: All access attempts logged (user, timestamp, action)
- [ ] Memory file permissions: OS-level file permissions (only authorized users can read/write)

Single-user mode: Access control optional (default: open access on localhost).

NFR-12: Audit & Compliance

Priority: P0 (Critical)

Requirements:
- [ ] **Immutable audit trail**: All decisions, actions, errors logged; logs are append-only
- [ ] **Timestamped entries**: All log entries include UTC timestamp
- [ ] **Searchable logs**: Logs in plaintext or JSON (grep-compatible)
- [ ] **Exportable reports**: Audit trail can be exported to PDF or HTML for compliance officers
- [ ] **Provenance tracking**: Every output includes: which model, why selected, what checks

performed, sources cited
- [ ] **Tamper detection**: Logs include checksums or signatures (detect unauthorized modifications)

NFR-13: Model Safety & Alignment

Priority: P1 (High)

Requirements:
- [ ] Validator checks outputs for: harmful content, policy violations, sensitive data leakage
- [ ] Safety policies configurable (e.g., no PII in outputs, no harmful instructions)
- [ ] Red-teaming test suite: 50+ adversarial prompts (prompt injection, jailbreak attempts)
- [ ] Safety violations logged and blocked before reaching user
- [ ] Option to enable/disable safety checks (for research vs. production)

4.5 Usability & Maintainability

NFR-14: Observability

Priority: P1 (High)

Requirements:
- [ ] Real-time monitoring dashboard (Grafana or equivalent)
- [ ] Key metrics visible: latency (p50/p95/p99), resource utilization (GPU/CPU/RAM), error rates, routing accuracy
- [ ] Alerting on critical conditions: OOM imminent, high error rate, validation bottleneck
- [ ] Log aggregation (if multi-node): Centralized logs with search capability
- [ ] Distributed tracing (optional): Track request flow across Router → Worker → Validator

---

NFR-15: Debuggability

Priority: P1 (High)

Requirements:
- [ ] Human-readable logs (not just machine logs)
- [ ] Markdown memory ledger is self-documenting (domain experts can inspect)
- [ ] Decision trails: Every routing/validation decision includes reasoning ("Why this model? Why rejected?")
- [ ] Verbose mode: CLI/API can request detailed logs (for troubleshooting)
- [ ] Error messages: Specific, actionable (not generic "Error 500")

NFR-16: Maintainability

Priority: P1 (High)

Requirements:
- [ ] Modular architecture: Router, Worker, Validator, Memory, Tools are separate, replaceable components
- [ ] Model updates: New models can be added without code changes (configuration-driven)

- [ ] Routing logic updates: Prolog rules can be edited independently of Python code
- [ ] Documentation: Comprehensive (architecture, API, operations runbook, troubleshooting)
- [ ] Code quality: Linted (Pylint, Black), type-hinted (Python), tested (unit + integration tests)

4.6 Scalability & Extensibility

NFR-17: Extensibility (Future-Proofing)

Priority: P2 (Nice-to-have for v1.0, required for future versions)

Requirements:
- [ ] Pluggable model interface: New specialist models can be added via configuration
- [ ] Pluggable tool interface: New tools (e.g., web search, calculators) can be added without core changes
- [ ] Multi-GPU support (future): Architecture supports scaling to multiple GPUs (not implemented in v1.0)
- [ ] Distributed deployment (future): Architecture supports splitting Router, Workers, Validator across multiple machines

---

NFR-18: Horizontal Scalability (Out of Scope for v1.0, documented for future)

Priority: P2 (Future)

Future requirements (not v1.0):
- [ ] Multi-user concurrent access: Support 10+ concurrent users
- [ ] Load balancing: Distribute requests across multiple Worker instances
- [ ] Stateless architecture: Enable horizontal scaling without shared state (except memory ledger)

---

5. User Stories (Detailed)

Epic 1: Code Generation & Validation

Story 1.1: As a Technical Lead, I want to generate validated production code so that I can trust AI outputs without extensive manual review.

Acceptance Criteria:
- [ ] I submit a coding request via CLI or API
- [ ] System routes to appropriate coding specialist (Qwen or Nemotron)
- [ ] System validates code incrementally (block-by-block for high-stakes)
- [ ] I receive validated code + validation report
- [ ] Validation report shows: what was checked, what passed, what was corrected
- [ ] I can audit the decision trail (why this model, why this validation approach)

Priority: P0
Estimated Effort: Covered by FR-1 to FR-5

Story 1.2: As a Technical Lead, I want to refactor legacy code with architectural validation so that refactorings are sound and complete.

Acceptance Criteria:
- [ ] I provide legacy code (multi-file codebase)
- [ ] System routes to Qwen Coder (architecture specialist)
- [ ] System generates refactoring plan (architecture-level changes)
- [ ] Validator reviews plan for: missing edge cases, breaking changes, structural issues
- [ ] I receive refactoring plan + validation report
- [ ] Plan is implementable (Nemotron can execute it)

Priority: P1
Dependencies: FR-12, FR-3

Epic 2: Document Analysis & Compliance

Story 2.1: As a Compliance Officer, I want to extract structured data from scanned medical records without data leaving premises so that I comply with HIPAA.

Acceptance Criteria:
- [ ] I upload scanned medical record (PDF/image)
- [ ] System runs OCR, extracts text with provenance (page numbers)
- [ ] System routes to GPT-OSS (reasoning/extraction)
- [ ] System validates extractions are grounded in OCR text (no hallucinations)
- [ ] I receive structured data (patient info, diagnoses, medications) + grounding citations
- [ ] I can verify every extracted field against source document
- [ ] Audit trail confirms data never transmitted externally

Priority: P0
Dependencies: FR-7a (OCR), FR-8 (Grounding), NFR-10 (Data Sovereignty)

Story 2.2: As a Compliance Officer, I want to generate audit reports showing that AI decisions are transparent and auditable so that I pass compliance audits.

Acceptance Criteria:
- [ ] I request audit report for a specific task (or date range)
- [ ] System exports audit trail: routing decisions, validation checks, grounding citations, timestamps
- [ ] Report format: PDF or HTML (auditor-friendly)
- [ ] Report includes: what models were used, why, what was checked, what passed/failed
- [ ] Auditors can verify decisions are deterministic and logged

Priority: P1
Dependencies: FR-10 (Audit Trail), NFR-12 (Audit & Compliance)

Epic 3: Multimodal Research

Story 3.1: As a Researcher, I want to analyze scientific papers with figures so that I can extract insights from both text and images.

Acceptance Criteria:
- [ ] I provide research paper (PDF with embedded images)
- [ ] System extracts text (OCR if needed) and processes figures (vision encoder)
- [ ] System generates research summary citing text paragraphs and figure descriptions
- [ ] I receive summary + citations (text + figures)
- [ ] I can trace every claim back to source (text or image)

Priority: P1
Dependencies: FR-7a (OCR), FR-7b (Vision), FR-8 (Grounding)

Story 3.2: As a Researcher, I want to retrieve relevant context from past experiments before generating new hypotheses so that I build on previous work.

Acceptance Criteria:
- [ ] I submit hypothesis generation request
- [ ] System retrieves relevant snippets from Markdown memory (past experiments, learned patterns)
- [ ] System uses retrieved context to inform hypothesis generation
- [ ] I receive hypothesis + references to past work
- [ ] I can see what prior knowledge informed the output

Priority: P2 (Nice-to-have)
Dependencies: FR-7c (Embeddings), FR-4 (Memory System)

Epic 4: Transparent Routing & Governance

Story 4.1: As a Technical Lead, I want to understand why the system chose a specific model for my request so that I can trust and debug routing decisions.

Acceptance Criteria:
- [ ] I submit a request
- [ ] System logs routing decision with reasoning
- [ ] I can view: “Routed to Qwen Coder (architecture) because: domain=coding, complexity=high, keywords=[refactor, multi-file]”
- [ ] Routing confidence score is shown (e.g., 92%)
- [ ] I can override routing decision if I disagree

Priority: P1
Dependencies: FR-2 (Routing), FR-10 (Audit Trail)

Story 4.2: As a Technical Lead, I want to configure validation policies for different task types so that I balance speed and rigor.

Acceptance Criteria:
- [ ] I configure validation policies in config file or via CLI

- [ ] Policies: low-stakes (no validation), medium-stakes (end-stage validation), high-stakes (block-by-block)
- [ ] System applies policy based on stakes assessment or my override
- [ ] I can see which policy was applied in audit trail

Priority: P1
Dependencies: FR-9 (Validation Policies), FR-20 (Configuration)

Epic 5: System Operations & Maintenance

Story 5.1: As a DevOps Engineer, I want to monitor system health in real-time so that I can detect and respond to issues quickly.

Acceptance Criteria:
- [ ] I access monitoring dashboard (Grafana)
- [ ] Dashboard shows: latency (p50/p95/p99), VRAM/RAM/CPU utilization, error rates, routing accuracy, validation rates
- [ ] Alerts trigger on critical conditions (OOM, high latency, high error rate)
- [ ] I receive alerts via email/Slack/PagerDuty

Priority: P1
Dependencies: NFR-14 (Observability), Phase 6 (Operations)

Story 5.2: As a DevOps Engineer, I want to update models without downtime so that I can deploy new versions seamlessly.

Acceptance Criteria:
- [ ] I download new model version, quantize, place in model vault
- [ ] I update configuration (model path, version)
- [ ] System reloads configuration (hot-reload)
- [ ] New model is used for subsequent requests
- [ ] In-progress tasks complete with old model (no interruption)

Priority: P2 (Nice-to-have for v1.0)
Dependencies: FR-20 (Configuration), Operations docs

---

6. Constraints & Assumptions

6.1 Constraints

Hardware Constraints (Fixed)

- **GPU**: NVIDIA Tesla A2 with 16GB VRAM (or equivalent; cannot exceed 16GB)
- **RAM**: 128GB ECC (minimum 64GB acceptable for PoC)
- **Storage**: 1TB NVMe SSD (minimum 500GB free)
- **CPU**: Intel Xeon W-2135 (6 cores) or AMD equivalent with AVX-512 support

Software Constraints

- **OS**: Linux (Ubuntu 22.04 LTS recommended); Windows/macOS out of scope for v1.0
- **Models**: Open-weight models only (no proprietary APIs)

- **Quantization**: GGUF format (llama.cpp); other formats out of scope

Operational Constraints

- **Single-user or small team**: Not designed for large-scale concurrent access (>5 users)
- **On-premise only**: Cloud deployment out of scope for v1.0
- **Sequential task execution**: No parallel task execution (one task at a time per Worker)

6.2 Assumptions

Technical Assumptions

- [ ] **Model availability**: Specified models (Qwen, Nemotron, GPT-OSS, MythoMax, Granite) are available for download
- [ ] **Quantization quality**: Q4/Q5 quantization preserves acceptable quality (validated during Phase 0)
- [ ] **CPU validation speed**: CPU can sustain ≥3 tokens/second for Granite-H-Small (validated during Phase 0)
- [ ] **PCIe bandwidth**: RAM → VRAM transfers are fast enough (<3 seconds for 12-18GB model)
- [ ] **Prolog availability**: SWI-Prolog or GNU Prolog available and performant for routing logic

User Assumptions

- [ ] **Technical proficiency**: Users are comfortable with CLI or API interfaces (no GUI required)
- [ ] **Latency tolerance**: Users accept 10-60 second task completion times (not real-time)
- [ ] **Quality over speed**: Users prioritize validated, reliable outputs over instant responses

Organizational Assumptions

- [ ] **Data sovereignty need**: Organization has genuine compliance requirements (HIPAA, GDPR, etc.)
- [ ] **Hardware availability**: Organization can procure or already owns specified hardware
- [ ] **Operational capability**: Organization has DevOps/SRE capability to deploy and maintain system

---

7. Dependencies & Risks

7.1 External Dependencies

Dependency	Type	Risk	Mitigation
Open-weight models	Model availability	Medium	Have alternative models identified (e.g., Llama, Mixtral alternatives)
llama.cpp	Inference engine	Low	Mature, widely-used; fallback: vLLM or other engines
SWI-Prolog	Routing logic	Low	Mature, stable; fallback: GNU Prolog or embed logic in Python
Hardware procurement	Hardware availability	Medium	Validate during Phase 0; have fallback specs (e.g., 24GB GPU, 64GB RAM)
Tesseract OCR	OCR engine	Low	Widely available; alternative: PaddleOCR
Sentence-Transformers	Embeddings	Low	Mature library; many model options available

7.2 Project Risks

Risk 1: Hardware Insufficient

Likelihood: Medium
Impact: High (project failure)
Mitigation:
- Validate hardware capabilities in Phase 0 (baseline testing)
- Have contingency models ready (smaller models, more aggressive quantization)
- Identify minimum acceptable hardware if current hardware fails

Risk 2: Router Accuracy Below Target

Likelihood: Medium
Impact: High (poor user experience)
Mitigation:
- Invest heavily in Phase 1 (routing logic)
- Iterative refinement based on test data
- Fallback: Embedding-based classification or hybrid approach
- Accept lower accuracy (75-80%) if explainability is high

Risk 3: Validation Adds Unacceptable Latency

Likelihood: Medium
Impact: High (system unusable)
Mitigation:
- Test early (Phase 2)
- Optimize validator prompts (reduce output verbosity)
- Make validation optional for low-stakes tasks
- Accept slower execution for high-stakes (user expectation management)

Risk 4: Complexity Overwhelms Team

Likelihood: Medium
Impact: High (delays, bugs, abandonment)
Mitigation:
- Start small (2-3 models in Phase 0-1)
- Add complexity only when justified (measured value)
- Rigorous documentation and knowledge sharing
- External consulting if needed (Prolog, llama.cpp experts)

Risk 5: Real-World Performance ≠ Lab Performance

Likelihood: High
Impact: Medium (production issues, user dissatisfaction)
Mitigation:
- Comprehensive monitoring from day one
- Gradual production rollout (canary deployment)
- Rapid iteration capability (fix issues quickly)
- Clear user expectations (this is v1.0, not perfect)

Risk 6: Model Quality Insufficient

Likelihood: Medium
Impact: High (outputs not trusted)
Mitigation:
- Validate model quality in Phase 0 (subjective assessment)
- Have alternative models identified and tested
- Prompt engineering (Phase 2, 5)
- Consider fine-tuning (future, out of scope for v1.0)

---

8. Success Metrics & KPIs

8.1 Technical KPIs

Metric	Target	Measurement Method	Frequency
Router Accuracy	≥90%	Test dataset (200+ prompts)	Weekly during dev, monthly in production
Validator False Positive Rate	<5%	Test dataset (100+ blocks)	Weekly during dev, monthly in production
Validator False Negative Rate	<3%	Test dataset (100+ blocks)	Weekly during dev, monthly in production
End-to-End Latency (p95)	≤60 seconds	Prometheus metrics	Real-time dashboard
Model Swap Time (p95)	≤5 seconds	Prometheus metrics	Real-time dashboard
System Uptime	≥99%	Uptime monitoring	Real-time dashboard
VRAM Utilization (max)	≤16GB	Prometheus metrics	Real-time dashboard
RAM Utilization (max)	≤120GB	Prometheus metrics	Real-time dashboard
OCR Accuracy	≥90%	Manual spot-checks	Monthly
Grounding Accuracy	≥80%	Manual spot-checks	Monthly

8.2 User Satisfaction KPIs

Metric	Target	Measurement Method	Frequency
User Satisfaction Score	≥7/10	Survey (1-10 scale)	Quarterly
Output Trust Score	≥7/10	Survey: “Do you trust outputs?”	Quarterly
Task Success Rate	≥85%	“Did output meet your needs?”	Per task (optional feedback)
Feature Requests	Track	User feedback channels	Ongoing
Bug Reports	<10/month	Issue tracker	Ongoing

8.3 Business/Operational KPIs

Metric	Target	Measurement Method	Frequency
Incidents (P0)	0	Incident tracking	Real-time
Incidents (P1)	<3/month	Incident tracking	Monthly review
Mean Time to Recovery (MTTR)	≤5 minutes	Incident logs	Per incident
Compliance Audit Pass Rate	100%	Audit results	Per audit (annual/quarterly)
Cost Savings vs. Cloud	Quantify	Cost analysis	Quarterly
Deployment Time (new instance)	≤4 hours	Deployment logs	Per deployment

---

9. Acceptance Criteria (High-Level)

The product is considered **ready for production** (v1.0 release) when:

9.1 Functional Acceptance

- [ ] All P0 functional requirements (FR-1 to FR-11, FR-16, FR-17) are implemented and tested
- [ ] All P0 specialist models (Router, Validator, Qwen, Nemotron, GPT-OSS) are operational

- [ ] Router accuracy ≥85% on test dataset
- [ ] Validator false positive <10%, false negative <5%
- [ ] Markdown memory system works reliably (no corruption, sessions resume)
- [ ] Multimodal pipelines (OCR, embeddings at minimum) functional
- [ ] Audit trail complete and exportable

9.2 Non-Functional Acceptance

- [ ] All P0 non-functional requirements (NFR-1 to NFR-7, NFR-10, NFR-12) are met
- [ ] System runs on specified hardware without OOM or crashes
- [ ] End-to-end latency ≤60 seconds (p95) for typical tasks
- [ ] System uptime ≥95% during staging deployment (1 week minimum)
- [ ] Data sovereignty verified (network monitoring confirms no external traffic)
- [ ] Disaster recovery tested (restore from backup successful)

9.3 Operational Acceptance

- [ ] All Phase 6 deliverables complete (monitoring, deployment automation, runbook)
- [ ] Monitoring dashboard operational (Grafana + Prometheus)
- [ ] Alerting working (test alerts triggered and received)
- [ ] Deployment can be executed by any team member following runbook
- [ ] All documentation complete (architecture, API, operations, user guide)

9.4 User Acceptance

- [ ] At least 3 real-world tasks completed successfully by end users
- [ ] User feedback collected (survey or interviews)
- [ ] No critical user-reported issues (P0 bugs)
- [ ] Users report satisfaction ≥6/10 (acceptable for v1.0)

9.5 Sign-Off

- [ ] Technical Lead approves (technical quality)
- [ ] Product Lead approves (requirements met)
- [ ] Operations Lead approves (operational readiness)
- [ ] Security Architect approves (security/compliance)
- [ ] End users approve (user acceptance testing)

---

10. Roadmap & Phasing

See separate document: Project_Roadmap.md (Project_Roadmap.md)

Summary:
- **Phase 0**: Foundation (4 weeks)
- **Phase 1**: Router (4 weeks)
- **Phase 2**: Validation (4 weeks)
- **Phase 3**: Full Stack (4 weeks)
- **Phase 4**: Multimodal (4 weeks)
- **Phase 5**: Optimization (4 weeks)
- **Phase 6**: Operations (4 weeks)

- **Phase 7**: Production (6 weeks)
- **Total**: 6-8 months to production

---

11. Future Roadmap (Post-v1.0)

v1.1: Usability Enhancements (3-6 months post-v1.0)

- [ ] Web-based GUI (dashboard for monitoring + task submission)
- [ ] Streaming outputs (token-by-token generation)
- [ ] Batch processing mode (submit multiple tasks)
- [ ] Improved prompt engineering tools (A/B testing UI)

v2.0: Multi-User & Scalability (6-12 months post-v1.0)

- [ ] Multi-user concurrent access (10+ users)
- [ ] Load balancing across multiple Worker instances
- [ ] Multi-GPU support (horizontal scaling)
- [ ] Distributed deployment (Router, Workers, Validator on separate machines)
- [ ] Cloud deployment option (private cloud, Kubernetes)

v2.5: Advanced Multimodal (12-18 months post-v1.0)

- [ ] Audio/video input support (transcription, visual analysis)
- [ ] Advanced vision capabilities (diagram understanding, visual reasoning)
- [ ] Multimodal generation (code + diagrams, reports + visualizations)

v3.0: Fine-Tuning & Customization (18-24 months post-v1.0)

- [ ] Fine-tuning capability (domain-specific adaptation)
- [ ] Custom validator training (organization-specific policies)
- [ ] Reinforcement learning from human feedback (RLHF)
- [ ] Active learning (system learns from user corrections)

---

12. Stakeholder Communication Plan

12.1 Stakeholder Matrix

Stakeholder	Interest	Influence	Communication Frequency	Preferred Channel
Product Lead	High	High	Daily	Slack, Weekly meetings
Technical Lead	High	High	Daily	Slack, Daily standups
Engineering Team	High	Medium	Daily	Slack, Standups, Sprint planning
DevOps/SRE	Medium	Medium	Weekly	Email, Bi-weekly sync
Security Architect	Medium	High	Bi-weekly	Email, Security reviews
End Users	High	Low	Monthly	Email updates, Quarterly surveys
Management	Medium	High	Monthly	Executive summary, Quarterly demos

12.2 Communication Deliverables

Deliverable	Audience	Frequency	Owner
Sprint Demo	All stakeholders	Bi-weekly	Technical Lead
Executive Summary	Management	Monthly	Product Lead
Technical Deep-Dive	Engineering	Monthly	Technical Lead
User Update	End users	Monthly	Product Lead
Incident Report	Management, Operations	As needed	On-call engineer
Quarterly Review	All stakeholders	Quarterly	Product Lead + Technical Lead

13. Open Questions & Decisions Pending

13.1 Technical Decisions

Question	Options	Decision Owner	Target Date	Status
Prolog implementation choice?	SWI-Prolog vs. GNU Prolog vs. embedded in Python	Technical Lead	Phase 1 start	Pending
Vector database for embeddings?	FAISS vs. ChromaDB vs. numpy	ML Lead	Phase 4 start	Pending
OCR engine choice?	Tesseract vs. PaddleOCR vs. commercial	ML Lead	Phase 4 start	Pending
Vision encoder model?	CLIP vs. BLIP vs. LLaVA	ML Lead	Phase 4 start	Pending
Monitoring stack?	Prometheus+Grafana vs. ELK vs. Datadog	DevOps Lead	Phase 6 start	Pending
Containerization?	Docker vs. bare metal deployment	DevOps Lead	Phase 6 start	Pending

13.2 Scope Decisions

Question	Impact	Decision Owner	Target Date	Status
Include MythoMax (creative) in v1.0?	Nice-to-have; adds complexity	Product Lead	Phase 3 start	Pending
Implement streaming outputs in v1.0?	High user value; significant effort	Product Lead	Phase 2 review	Pending
GUI in v1.0 vs. defer to v1.1?	Usability vs. timeline	Product Lead	Before Phase 3	Pending
Multi-validator approach (parallel validation)?	Higher accuracy; higher latency	Technical Lead	Phase 5	Pending

---

14. Glossary

Term	Definition
Bicameral Architecture	Design separating creative generation (GPU) from critical validation (CPU), mimicking biological cognition (creative + critical hemispheres)
Grounding	Ensuring AI outputs are attributable to source material (OCR text, image features); preventing hallucinations
GGUF	File format for quantized models compatible with llama.cpp
KV Cache	Key-Value cache storing attention computation results for faster inference
MoE (Mixture-of-Experts)	Model architecture using multiple specialized sub-networks, activating only a subset per input for efficiency
Provenance	Tracking the source and lineage of data (which document, which page, which model generated it)
Quantization	Reducing numerical precision (e.g., FP32 → INT4) to decrease model size with acceptable quality loss
Sovereign AI	Local, privacy-preserving AI systems with no external dependencies; full user control
Stakes	Risk level of a task (low, medium, high); determines validation rigor
Validator Ladder	Sequential validation architecture with multiple checking stages (line-by-line, block-by-block, stage-by-stage)
Warm Pool	Strategy keeping likely-next models in RAM for faster GPU loading vs. disk loading
Worker	Specialist AI model performing creative generation (coding, reasoning, creative writing)

15. Appendices

Appendix A: References

1. **Technical Analysis Report** (Deep Agent Report.pdf) - Comprehensive architecture analysis
2. **Hybrid Logic Router Report** (Comprehensive Report - Hybrid Logic Router.pdf) - Final iteration architecture
3. **Project Roadmap** (Project_Roadmap.md) - Phased implementation timeline
4. **Document Roadmap** (Document_Roadmap.md) - Full documentation plan

Appendix B: Comparison with Alternatives

Approach	Pros	Cons	Verdict
Cloud APIs (OpenAI, Anthropic)	Powerful, maintained, latest models	No data sovereignty, ongoing costs, no transparency	❌ Unacceptable (data privacy)
Single local model (Llama 70B)	Simple, fast, no orchestration	No specialization, no validation, less capability	❌ Insufficient (no governance)
Multiple models, no validation	Specialization benefits	No quality governance, still has hallucinations	❌ Incomplete (no validation)
This solution (Bicameral + Validation)	Sovereignty + governance + specialization + transparency	Complex, slower, requires hardware	✅ **Recommended** (meets all requirements)

Appendix C: Cost-Benefit Analysis

Benefits (qualitative):
- **Data sovereignty**: Enables AI use in regulated industries (healthcare, finance, legal, defense)
- **Quality governance**: Reduces manual review burden, increases trust in outputs
- **Transparency**: Supports compliance audits, debugging, user trust
- **Cost savings**: No ongoing API costs (vs. $100-1000+/month for cloud APIs)
- **Control**: Full customization, no vendor lock-in, no censorship

Costs:
- **Hardware**: $3,000-5,000 (one-time; may already be available)
- **Development**: 3-5 engineers × 6-8 months = $200k-500k (highly variable)
- **Ongoing operations**: 0.5-1 FTE for maintenance = $50k-150k/year
- **Complexity**: Higher operational burden than cloud APIs

ROI Calculation (example for 10-person team using cloud APIs):
- **Cloud API costs**: $200/person/month × 10 people × 12 months = $24,000/year
- **Break-even**: System pays for itself in 1-2 years (vs. cloud APIs) if development costs are amortized
- **Intangible benefits**: Data sovereignty (priceless for regulated industries), trust, transparency

Verdict: **High ROI for organizations with data sovereignty requirements**; questionable ROI for general-purpose use (cloud APIs may be simpler).

Document Approval & Sign-Off

Product Lead: _____ Date: _

Technical Lead: _____ Date: _

Engineering Manager: _____ Date: _

Security Architect: _____ Date: _

Operations Lead: _____ Date: _

PRD Version: 1.0
Last Updated: February 5, 2026
Next Review: End of Phase 2 (Week 12)
Maintained By: Product Lead / Technical Lead

End of Product Requirements Document