Document Roadmap: Sovereign AI Infrastructure

Bicameral Validator Ladder with Hybrid Logic Router

Project: Local Multi-Model AI System with GPU/CPU Heterogeneous Compute
Version: 1.0
Date: February 5, 2026

Purpose of This Roadmap

This document roadmap guides you through **every document you need to create** from initial concept to operational deployment. As your first project, this roadmap ensures you don't miss critical planning, design, or operational documentation.

Documents are organized in **sequential phases**. Each document builds on previous ones and feeds into the next. Dependencies are clearly marked.

Phase 0: Foundation & Planning Documents

0.1 ✅ Technical Analysis Report (COMPLETED)

- **Status**: Already exists (Deep Agent Report.pdf)
- **Purpose**: Comprehensive analysis of the architecture, critical evaluation, feasibility
- **Key Sections**: Architecture components, validation approach, hardware constraints
- **Audience**: Technical team, stakeholders
- **Dependencies**: None
- **Outputs to next phase**: Validated technical approach, identified risks, component specifications

0.2 ✅ Comprehensive Architecture Report (COMPLETED)

- **Status**: Already exists (Comprehensive Report - Hybrid Logic Router.pdf)
- **Purpose**: Final iteration architecture with Prolog/Python hybrid, OCR/vision/embeddings
- **Key Sections**: Bicameral design, governance policies, implementation blueprint
- **Audience**: Implementation team
- **Dependencies**: Technical Analysis Report
- **Outputs to next phase**: Detailed architecture, expanded capability requirements

0.3 📄 Product Requirements Document (PRD) ← YOU ARE HERE

- **Status**: In progress
- **Purpose**: Define what you're building, why, for whom, and success criteria
- **Key Sections**: Product vision, user stories, functional/non-functional requirements, success metrics
- **Audience**: All stakeholders, implementation team, future maintainers

- **Dependencies**: Both analysis reports
- **Outputs to next phase**: Clear requirements baseline, acceptance criteria, scope boundaries
- **Estimated effort**: 3-5 days
- **Owner**: Product Lead / Technical Lead

Phase 1: Design & Specification Documents

1.1 📄 System Architecture Document (SAD)

- **Purpose**: Translate PRD requirements into detailed technical architecture
- **Key Sections**:
- Component diagrams (GPU/CPU split, model placement)
- Data flow diagrams (Generate → Validate → Commit loop)
- Interface specifications (Python ↔ Prolog, model API contracts)
- Technology stack details (llama.cpp, SWI-Prolog, vector DB choice)
- Deployment architecture
- **Audience**: Software engineers, DevOps
- **Dependencies**: PRD, Architecture Reports
- **Outputs to next phase**: Technical blueprints for implementation
- **Estimated effort**: 1-2 weeks
- **Owner**: Solutions Architect / Technical Lead
- **Artifacts**: Architecture diagrams (C4 model recommended), API specs, data models

1.2 📄 Data Architecture & Memory Design Document

- **Purpose**: Define the Markdown memory bus, knowledge graph structure, persistence layer
- **Key Sections**:
- Markdown file schemas (project_state.md, scratchpad.md, knowledge_graph.md)
- Vector database schema (embeddings storage)
- Provenance tracking design (OCR sources, vision input metadata)
- Backup and versioning strategy
- Concurrency control mechanisms
- **Audience**: Backend engineers, data engineers
- **Dependencies**: PRD, System Architecture Document
- **Outputs to next phase**: Memory system implementation specs
- **Estimated effort**: 1 week
- **Owner**: Data Architect / Backend Lead

1.3 📄 Routing Logic Specification (Prolog Predicates)

- **Purpose**: Define declarative routing rules, validation policies, grounding requirements
- **Key Sections**:
- Predicate definitions (routing rules, validation policies, tool selection)
- Decision tree examples (given input X, why model Y was chosen)
- Stakes classification logic (low/medium/high)
- Uncertainty handling rules
- Grounding policy constraints (OCR/vision source-backing)

- **Audience**: Logic programming developers, system architects
- **Dependencies**: PRD, System Architecture Document
- **Outputs to next phase**: Prolog implementation roadmap
- **Estimated effort**: 1-2 weeks
- **Owner**: Logic Systems Engineer / Routing Lead

1.4 📄 Model Serving & Orchestration Design

- **Purpose**: Define how models are loaded, managed, and orchestrated
- **Key Sections**:
- Model loading/unloading sequences
- Warm pool management algorithm
- GPU ↔ RAM transfer protocols
- Quantization strategy per model
- Failure recovery (OOM handling, model swap failures)
- Monitoring and telemetry points
- **Audience**: ML engineers, DevOps
- **Dependencies**: System Architecture Document
- **Outputs to next phase**: Orchestration implementation specs
- **Estimated effort**: 1 week
- **Owner**: ML Infrastructure Engineer

1.5 📄 Multimodal Pipeline Design Document

- **Purpose**: Specify OCR, vision encoder, and embedding model integration
- **Key Sections**:
- OCR pipeline (input → extraction → provenance → output)
- Vision encoder pipeline (image → features → captions → embeddings)
- Embedding model pipeline (text → vector → retrieval)
- Tool selection routing (when to invoke each pipeline)
- Error handling and confidence thresholds
- **Audience**: ML engineers, backend engineers
- **Dependencies**: System Architecture Document, Routing Logic Specification
- **Outputs to next phase**: Multimodal tool implementation specs
- **Estimated effort**: 1 week
- **Owner**: Multimodal Systems Engineer

1.6 📄 Security & Governance Design Document

- **Purpose**: Define security controls, audit mechanisms, governance policies
- **Key Sections**:
- Threat model (prompt injection, model manipulation, memory tampering)
- Access controls (who can invoke which models/tools)
- Audit logging requirements (decision trails, provenance)
- Data privacy measures (local-only guarantees, encryption at rest)
- Compliance considerations (if applicable)
- **Audience**: Security engineers, compliance officers
- **Dependencies**: PRD, System Architecture Document
- **Outputs to next phase**: Security implementation requirements

- **Estimated effort**: 1 week
- **Owner**: Security Architect

Phase 2: Implementation Documents

2.1 📄 Development Plan & Sprint Structure

- **Purpose**: Break down implementation into manageable sprints
- **Key Sections**:
- Sprint breakdown (aligned with phased approach from reports)
- Team structure and role assignments
- Development environment setup instructions
- Code repository structure
- Branching and merge strategy
- Definition of Done for each sprint
- **Audience**: Development team, project manager
- **Dependencies**: All Phase 1 design documents
- **Outputs to next phase**: Sprint backlogs, resource allocation
- **Estimated effort**: 3-5 days
- **Owner**: Engineering Manager / Scrum Master

2.2 📄 API Specification Document

- **Purpose**: Define all internal and external API contracts
- **Key Sections**:
- Python orchestrator API (REST/gRPC endpoints)
- Prolog query interface (Python ↔ Prolog protocol)
- Model server APIs (llama.cpp endpoints)
- Tool APIs (OCR, vision, embeddings)
- Markdown memory bus access patterns
- Error response schemas
- **Audience**: Full-stack engineers, integration developers
- **Dependencies**: System Architecture Document
- **Outputs to next phase**: OpenAPI specs, integration test requirements
- **Estimated effort**: 1 week
- **Owner**: API Lead / Backend Architect
- **Artifacts**: OpenAPI/Swagger specs, Postman collections

2.3 📄 Testing Strategy & Test Plan

- **Purpose**: Define comprehensive testing approach
- **Key Sections**:
- Unit testing strategy (per component)
- Integration testing approach (router → model → validator flows)
- End-to-end testing scenarios (50+ representative tasks)
- Performance testing (latency, throughput, resource utilization)
- Validation accuracy testing (false positive/negative rates)

- Multimodal pipeline testing (OCR/vision accuracy)
- Load testing (GPU/CPU/RAM stress scenarios)
- Security testing (prompt injection, bypass attempts)
- **Audience**: QA engineers, developers
- **Dependencies**: All Phase 1 design documents
- **Outputs to next phase**: Test cases, test data sets, acceptance criteria
- **Estimated effort**: 1 week
- **Owner**: QA Lead / Test Architect

2.4 📄 Model Prompt Engineering Guide

- **Purpose**: Document system prompts for each model role
- **Key Sections**:
- Worker prompts (per specialist: coding, reasoning, creative)
- Validator prompts (governance, structure checking, grounding verification)
- Router prompts (classification, uncertainty handling)
- Few-shot examples for each role
- Prompt versioning and A/B testing strategy
- Failure mode prompts (retry, clarification, escalation)
- **Audience**: ML engineers, prompt engineers
- **Dependencies**: Routing Logic Specification, Model Serving Design
- **Outputs to next phase**: Production-ready prompts, prompt templates
- **Estimated effort**: 2 weeks (iterative refinement expected)
- **Owner**: Prompt Engineering Lead

Phase 3: Deployment & Operations Documents

3.1 📄 Infrastructure Setup Guide

- **Purpose**: Step-by-step instructions to provision and configure hardware
- **Key Sections**:
- Hardware procurement checklist (Tesla A2, Xeon, RAM, NVMe)
- Operating system installation and configuration
- Driver installation (NVIDIA CUDA, cuDNN)
- Dependency installation (llama.cpp, SWI-Prolog, Python packages)
- Filesystem layout (model vault directory structure)
- Network configuration (if multi-node)
- **Audience**: DevOps engineers, system administrators
- **Dependencies**: System Architecture Document, Model Serving Design
- **Outputs to next phase**: Operational infrastructure
- **Estimated effort**: 3-5 days
- **Owner**: DevOps Lead

3.2 📄 Deployment Runbook

- **Purpose**: Detailed deployment procedures for production
- **Key Sections**:

- Pre-deployment checklist (model downloads, quantization, verification)
- Deployment sequence (services startup order)
- Configuration management (environment variables, config files)
- Health checks and smoke tests
- Rollback procedures
- Blue-green deployment strategy (if applicable)
- **Audience**: DevOps engineers, SREs
- **Dependencies**: Infrastructure Setup Guide
- **Outputs to next phase**: Repeatable deployment process
- **Estimated effort**: 1 week
- **Owner**: DevOps Lead / SRE

3.3 📄 Monitoring & Observability Plan

- **Purpose**: Define what to measure and how to monitor system health
- **Key Sections**:
- Metrics catalog (router accuracy, validation rates, latency, resource utilization)
- Logging strategy (structured logs, audit trails)
- Dashboard designs (Grafana dashboards)
- Alerting rules (OOM warnings, model swap failures, validation bottlenecks)
- Tracing instrumentation (distributed tracing for request flows)
- Performance baselines and SLOs
- **Audience**: SREs, DevOps, operations team
- **Dependencies**: System Architecture Document, Testing Strategy
- **Outputs to next phase**: Monitoring infrastructure, alert definitions
- **Estimated effort**: 1 week
- **Owner**: SRE Lead / Observability Engineer
- **Artifacts**: Prometheus configs, Grafana dashboards, alert rules

3.4 📄 Operations Manual (Runbook)

- **Purpose**: Day-to-day operational procedures
- **Key Sections**:
- Routine maintenance tasks (model updates, log rotation)
- Troubleshooting guide (common failure modes and resolutions)
- Performance tuning procedures
- Backup and recovery procedures
- Incident response protocols
- On-call procedures
- Capacity planning guidelines
- **Audience**: Operations team, on-call engineers
- **Dependencies**: Monitoring Plan, Deployment Runbook
- **Outputs to next phase**: Operational readiness
- **Estimated effort**: 1-2 weeks
- **Owner**: Operations Lead / SRE

3.5 📄 Disaster Recovery & Business Continuity Plan

- **Purpose**: Ensure system can recover from failures

- **Key Sections**:
- Backup strategy (models, memory files, configuration)
- Recovery time objectives (RTO) and recovery point objectives (RPO)
- Failure scenarios and recovery procedures (hardware failure, corruption, etc.)
- Failover mechanisms (if applicable)
- Data integrity verification procedures
- **Audience**: SREs, operations team, management
- **Dependencies**: System Architecture Document, Operations Manual
- **Outputs to next phase**: Disaster recovery capability
- **Estimated effort**: 3-5 days
- **Owner**: SRE Lead / Operations Manager

Phase 4: Knowledge Transfer & Maintenance Documents

4.1 📄 Developer Onboarding Guide

- **Purpose**: Enable new developers to contribute effectively
- **Key Sections**:
- System overview and architecture walkthrough
- Development environment setup
- Code repository navigation
- Key design patterns and conventions
- How to add a new model to the stack
- How to modify routing logic
- How to extend multimodal pipelines
- Debugging techniques and tools
- **Audience**: New developers, contractors
- **Dependencies**: All implementation and operations documents
- **Outputs to next phase**: Self-sufficient development team
- **Estimated effort**: 1 week
- **Owner**: Technical Lead / Senior Engineer

4.2 📄 User Guide / End-User Documentation

- **Purpose**: Teach end users how to interact with the system
- **Key Sections**:
- System capabilities overview
- How to submit requests (interface documentation)
- Understanding routing decisions (transparency features)
- Interpreting validation reports
- Working with OCR/vision inputs
- Best practices for high-quality outputs
- Limitations and known issues
- FAQ and troubleshooting

- **Audience**: End users, domain experts
- **Dependencies**: PRD, all design documents
- **Outputs to next phase**: User adoption
- **Estimated effort**: 1 week
- **Owner**: Technical Writer / Product Manager
- **Artifacts**: User manual (HTML/PDF), video tutorials, quick-start guides

4.3 📄 Model Update & Maintenance Guide

- **Purpose**: Document how to update models, prompts, and routing logic
- **Key Sections**:
- Model evaluation and selection criteria
- Quantization procedures
- Model integration checklist
- Prompt update and testing procedures
- Routing logic modification guidelines
- Regression testing requirements
- Performance impact assessment
- **Audience**: ML engineers, operations team
- **Dependencies**: Model Serving Design, Prompt Engineering Guide
- **Outputs to next phase**: Sustainable maintenance process
- **Estimated effort**: 3-5 days
- **Owner**: ML Lead / Operations Lead

4.4 📄 Lessons Learned & Post-Deployment Review

- **Purpose**: Capture insights for future improvements
- **Key Sections**:
- What worked well
- What didn’t work as expected
- Performance vs. projections (latency, accuracy, resource usage)
- Architectural decisions revisited
- Recommendations for next iteration
- Technical debt inventory
- **Audience**: All stakeholders, future teams
- **Dependencies**: Operational data, team retrospectives
- **Outputs to next phase**: Improvement roadmap
- **Estimated effort**: 1 week (after 3-6 months operation)
- **Owner**: Technical Lead / Product Manager

Phase 5: Continuous Improvement Documents (Ongoing)

5.1 📄 Performance Optimization Reports (Recurring)

- **Purpose**: Document optimization efforts and results
- **Cadence**: Quarterly or after major optimizations

- **Key Sections**: Identified bottlenecks, optimization approach, results, benchmarks

5.2 📄 Incident Post-Mortems (As Needed)

- **Purpose**: Learn from failures and prevent recurrence
- **Trigger**: Any significant outage or issue
- **Key Sections**: Timeline, root cause, impact, remediation, prevention

5.3 📄 Model Performance Evaluations (Recurring)

- **Purpose**: Assess whether current models still meet requirements
- **Cadence**: Quarterly or when new models released
- **Key Sections**: Current performance, new model candidates, evaluation results, recommendation

5.4 📄 Roadmap Updates (Recurring)

- **Purpose**: Evolve the system based on learnings and new requirements
- **Cadence**: Bi-annually or after major milestones
- **Key Sections**: Completed features, upcoming features, architectural evolution plans

---

Document Dependencies Visualization

```
Phase 0: Foundation
  ├─ Technical Analysis Report (✅)
  ├─ Architecture Report (✅)
  └─ PRD (📄 IN PROGRESS)
        ↓
Phase 1: Design
  ├─ System Architecture Document ← depends on PRD
  ├─ Data Architecture ← depends on PRD, SAD
  ├─ Routing Logic Spec ← depends on PRD, SAD
  ├─ Model Serving Design ← depends on SAD
  ├─ Multimodal Pipeline Design ← depends on SAD, Routing Spec
  └─ Security & Governance ← depends on PRD, SAD
        ↓
Phase 2: Implementation
  ├─ Development Plan ← depends on all Phase 1 docs
  ├─ API Specification ← depends on SAD
  ├─ Testing Strategy ← depends on all Phase 1 docs
  └─ Prompt Engineering Guide ← depends on Routing Spec, Model Serving
        ↓
Phase 3: Deployment & Ops
  ├─ Infrastructure Setup ← depends on SAD, Model Serving
  ├─ Deployment Runbook ← depends on Infrastructure Setup
  ├─ Monitoring Plan ← depends on SAD, Testing Strategy
  ├─ Operations Manual ← depends on Monitoring, Deployment
  └─ DR/BC Plan ← depends on SAD, Operations Manual
        ↓
Phase 4: Knowledge Transfer
  ├─ Developer Onboarding ← depends on all previous docs
  ├─ User Guide ← depends on PRD, all design docs
  ├─ Model Maintenance Guide ← depends on Model Serving, Prompts
  └─ Lessons Learned ← depends on operational data
        ↓
Phase 5: Continuous (ongoing)
  ├─ Performance Optimization Reports
  ├─ Incident Post-Mortems
  ├─ Model Performance Evaluations
  └─ Roadmap Updates
```

Document Ownership Matrix

Phase	Document	Primary Owner	Collaborators	Review Required From
0	Technical Analysis	✅ Complete	-	-
0	Architecture Report	✅ Complete	-	-
0	**PRD**	**Product Lead**	**Tech Lead, Stakeholders**	**All**
1	System Architecture	Solutions Architect	Dev Team	Tech Lead, Security
1	Data Architecture	Data Architect	Backend Lead	Tech Lead, Dev Team
1	Routing Logic Spec	Logic Systems Engineer	Architect	Tech Lead, ML Lead
1	Model Serving Design	ML Infrastructure Eng	DevOps	ML Lead, Architect
1	Multimodal Pipeline	Multimodal Systems Eng	ML Team	ML Lead, Architect
1	Security & Governance	Security Architect	Legal/Compliance	Tech Lead, Management
2	Development Plan	Engineering Manager	Scrum Master	Tech Lead
2	API Specification	API Lead	Backend Team	Architect, Dev Team
2	Testing Strategy	QA Lead	QA Team, Dev Team	Tech Lead, QA Manager
2	Prompt Engineering	Prompt Engineering Lead	ML Team	ML Lead, Tech Lead
3	Infrastructure Setup	DevOps Lead	SRE Team	Tech Lead, Operations
3	Deployment Runbook	DevOps Lead	SRE	Operations Manager
3	Monitoring Plan	SRE Lead	DevOps	

Phase	Document	Primary Owner	Collaborators	Review Required From
				Operations Manager
3	Operations Manual	Operations Lead	SRE Team	Operations Manager
3	DR/BC Plan	SRE Lead	Operations	Operations Manager, Management
4	Developer Onboarding	Technical Lead	Senior Engineers	Engineering Manager
4	User Guide	Technical Writer	Product Manager	Product Lead, UX
4	Model Maintenance	ML Lead	Operations	ML Manager, DevOps
4	Lessons Learned	Tech Lead / PM	All Teams	Management, Stakeholders

Critical Success Factors for Documentation

1. Documentation-Driven Development

- Write design docs BEFORE code
- All major architectural decisions must be documented and reviewed
- PRD is the contract; all work validates against it

2. Version Control

- All documents in Git (Markdown preferred for diff-ability)
- Document version tied to software version
- Change logs maintained for major documents

3. Review Process

- All Phase 1 (Design) docs require architecture review
- All Phase 2 (Implementation) docs require peer review
- All Phase 3 (Operations) docs require ops team sign-off

4. Living Documents

- Documents are never “done”—update as system evolves
- Quarterly review of all Phase 3 (Operations) documents
- Incident learnings feed back into runbooks immediately

5. Accessibility

- Central documentation repository (wiki or Git)
- Search capability across all documents
- Clear naming conventions and folder structure

Recommended Documentation Tools

For Diagramming:

- **Draw.io** / **Lucidchart**: Architecture diagrams
- **Mermaid**: Embedded diagrams in Markdown (Git-friendly)
- **C4 Model**: For system architecture (Context, Container, Component, Code)

For Documentation:

- **Markdown + Git**: Version-controlled, diff-friendly, accessible
- **Confluence** / **Notion**: If team prefers wiki-style
- **Docusaurus** / **MkDocs**: For static site generation from Markdown

For API Specs:

- **OpenAPI 3.0** / **Swagger**: REST API specifications
- **Protobuf**: If using gRPC

For Collaboration:

- **GitHub/GitLab Issues**: Track documentation tasks
- **Miro** / **FigJam**: Collaborative whiteboarding
- **Google Docs**: For draft reviews before committing to Git

---

Next Steps

1. ✅ **You are here**: Reviewing Document Roadmap
2. 📄 **Next**: Complete PRD (below this document)
3. 📄 **Then**: Begin Phase 1 (Design Documents), starting with System Architecture Document
4. 🔄 **Ongoing**: Maintain document dependencies and update this roadmap as you progress

---

Document Maintenance Schedule

Document Type	Review Frequency	Update Trigger
PRD	Quarterly	Scope changes, major feature requests
Architecture Docs	Bi-annually	Major architectural changes
API Specs	Per release	API changes
Operations Docs	Monthly	Process improvements, incid-ents
User Guide	Per release	Feature additions, UI changes
Monitoring Plan	Quarterly	New metrics, SLO changes
Security Docs	Quarterly	Threat model changes, audits

Document maintained by: Technical Lead / Product Manager
Last updated: February 5, 2026
Next review: May 5, 2026