SPEC-11-Sprite Light Assistants and Dynamic Model Orchestration

Background

Hearthlink requires intelligent resource utilization across devices ranging from high-performance desktops to minimal compute nodes. This specification began as a low-cost assistant layer (Sprites) but now extends into Core, Persona, Voice, and Sentry systems to enable:

- Lightweight task execution via 2B-class models
- Model orchestration for hot-swapping 7–8B reasoning engines
- Power-budget enforcement and adaptive workload scheduling

Requirements

Must Have

Fallback mechanism if model fails to load or exceeds power threshold – critical for non-disruptive user flow.

Audit log of model load/swap events with timestamps and session context – needed for debug and governance.

Should Have

- Conflict-resolution logic when model pool overlaps → Model conflict resolution via load order + LRU eviction policy – makes the outcome clearer.
- Parent-persona continuity layer – resumes last memory state when model hot-swap completes. Prevents UX stutter.

Could Have

- Wake-word support is useful but not MVP-aligned. Keep as-is.
- Shared Sprite access via network peer risks conflict with your security model (can be deferred or revised).

Won’t Have

- Cloud inference
- Assumed Offline Behavior

All Sprite and Persona functionality is expected to operate without any active internet connection. Model swaps, voice triggers, and telemetry defer gracefully until reconnected.

Method

Security Boundary for Model Isolation\ Each persona model (e.g., Llama 3 8B, Mistral 7B) is fully unloaded before loading a new one. Memory is cleared and GPU context reset to prevent residual context or memory bleed. This isolation ensures deterministic task routing and audit-safe handoff between models.

Scope and Impact

This specification affects multiple Hearthlink subsystems:

- **Sprite Light Assistants** – Five lightweight 2B model helpers
- **Primary Personas** – Gain hot-swap logic for model transitions (e.g., LLaMA 3 ⇄ Qwen ⇄ Mistral)
- **Voice System** – Gains universal Sprite triggering interface
- **Core Orchestrator** – Manages load, memory, and task queueing
- **Sentry** – Observes model activation, power, and thermal metrics
- If a Sprite call fails or returns below-threshold confidence, the parent Persona automatically escalates the task to the active reasoning model, maintaining user continuity without re-prompting.

Sprite Roster

| Sprite ... }] }

1. Implement Sprite orchestration layer in Core
2. Bind Voice protocol handlers to Sprite call interface
3. Add model preload + eviction logic (swap pool)
4. Extend settings schema and frontend to manage model behavior
5. Integrate power draw polling via NVIDIA-SMI / OS sensors
6. Add metrics forwarder to Sentry telemetry layer

Milestones

Week	Deliverable
1	Core Sprite engine w/ model routing
2	Voice-triggered task execution w/ confidence filter
3	Sentry integration for telemetry and warnings
4	UI tab for Sprite + Persona management
5	Dynamic model swap (8B class) + benchmark hooks
6	Power budget enforcement and test coverage

Gathering Results

- Sprite calls tracked in audit and telemetry with latency and accuracy scores

• Model swap logs cross-reference with GPU/CPU use to confirm benefit
• Energy dashboard plots 24h trend vs user thresholds
• User feedback UI (Settings > Feedback) enabled for Sprite scoring