**Doogie MVP Definition & Product Roadmap**

**MVP Goal**

Build a working multi-agent conversational AI prototype that:

* Uses LLMs to take structured, safe, explainable conversations
* Has observability, memory and prompt control
* Can be extended across verticals with different toolchains and prompts
* Is testable, measurable and designed for iterative improvement based on real and synthetic user feedback
* Is demo-able and designed to excite investors, partners and clinicians

The MVP should:

* Demonstrate real-time agent collaboration
* Store and recall longitudinal session data
* Support HITL (Human-in-the-Loop) evaluation
* Be agnostic to domain (extendable beyond medicine)
* Enable rapid iteration cycles with synthetic testing
* Lay a foundation that can be extended to other verticals (e.g. sales, support) with minimal architectural changes

Note: While this MVP focuses on the clinical domain, architectural decisions will be made to ensure future extensibility (e.g., modular prompts, pluggable tools, and abstracted agent definitions).

**Architecture Principles**

* LLM-Agnostic by Design: All agents use version-controlled prompts, metadata logging, and model-switch abstractions. MVP runs on a single LLM (e.g., GPT-4o or Claude), but the system allows for later flexibility.

* Observable by Design: Real-time and retrospective visibility into agent/tool behavior.

* Composable Agents: Modular agent roles with prompt and tool plug-ins.

* Tool-Augmented: External tools like NICE CKS, NHS dm+d, and web search are supported.

* Memory-Enabled: Long- and short-term memory for both patient context and agent reasoning.

* Testable by Default: Includes testing harnesses, synthetic persona simulation, and replay evaluation to support learn-fast iterations.

* Extensible by Intent: While the MVP focuses on clinical workflows, the system will be designed in a modular way to allow vertical swap-ins (via configuration, prompts, and pluggable APIs).

**Core MVP Features**

1. Chat + Prompt Interface

* Chat UI with metadata capture (LLM used, prompt version)
* Prompt controller (version-controlled, editable)
* Real-time and retrospective view in a combined dashboard

2. Agent Orchestration

* Modular agent engine (LLM + tool invocations)
* Each agent has a discrete responsibility
* Invocation trace and metadata per agent

3. Observability Layer

* Logs for agent events, tool use, prompt versions, outputs, latency
* Clinician / QA interface with real-time view of the patient-agent conversation and system actions
* Replay mode for evaluations and QA scoring

4. Context Memory

* FHIR-compatible synthetic patient loader (or extensible JSON schema for other verticals)
* Redis or vector database for short- and long-term memory
* Persistent context loading per session

5. Toolchain Integration (Pluggable)

* NICE CKS, SNOMED CT, ICD-10, NHS dm+d for health use case
* Open web search, retrieval augmented generation (RAG), or external APIs for general use
* Model integrations: Claude, GPT-4o or open-source LLMs

6. Testing Harness

* LLM-generated synthetic users based on medical case studies
* Automated test runs to evaluate whether the system asks the correct questions, collects complete data, and produces high-quality structured outputs