Goal:

To collaboratively build the six essential components of the SHIVA (Smart Hub for Intelligent Virtual Agents) platform.

This forms the foundation for an AI agents management system involving team tools, memory, and oversight. At the end of

the hackathon, teams will assemble the best implementation of each component into a functioning platform.

Model: The 'Core + Flex' Model

Each team must select one focus component as the primary focus and optionally choose a secondary component for contribution.

Components Overview:

1. Project "Manager" (Orchestrator)

   - Mission: Acts as the AI team lead. Receives high-level tasks from humans and delegates work to AI workers.
   - Key Tasks:
     * Translate mission into actionable plans
     * Build interface for user task approval and replanning
     * Communicate with Guardian for validation and Partner for execution
   - Technologies: Python (FastAPI/Flask), Node.js (Express)
   - Challenges: Async API state management, error handling, approval logic

2. Project "Partner" (The ReAct Runtime)

   - Mission: Acts as the autonomous worker executing delegated tasks using the ReAct framework.
   - Key Tasks:

  * Implement ReAct execution loop

  * Integrate with LLM and Guardian for validation

  * Connect with Resource Hub for tools and memory access

 - Technologies: Python (LangChain/LangGraph), FastAPI

 - Challenges: Reliable step logic, deviation detection, LLM validation

3. Project "Guardian" (The Compliance Assistant)

 - Mission: Dedicated compliance and safety partner ensuring security and rule-based approvals.

 - Key Tasks:

  * Implement policy validation logic

  * Validate plans and actions

  * Integrate with Resource Hub for policy context

 - Technologies: Python (FastAPI/Flask), Node.js (Express)

 - Challenges: Accurate rule enforcement and validation reporting

4. Project "Resource Hub" (The Armory, Library & Memory)

 - Mission: Central repository for capabilities, knowledge, and memory for all AI agents.

 - Key Tasks:

  * Build endpoints for RAG, tools, short-term and long-term memory

  * Implement secure sandboxed execution

 - Technologies: FastAPI, LangChain, Redis, ChromaDB

 - Challenges: Efficient memory structure, tool sandboxing

5. Project "Overseer" (The Observability Control Panel)

 - Mission: Monitoring and control hub providing real-time logs and kill-switch for agents.

 - Key Tasks:

  * Implement /log/event and /control/kill endpoints

    * Build simple web dashboard (React/HTML)

  - Technologies: FastAPI + WebSockets, Node.js + Socket.IO

  - Challenges: Real-time UI updates, stable control flow

6. Project "Directory" (The Service Registry)

  - Mission: Central address book for SHIVA platform. Handles service registration/discovery.

  - Key Tasks:

    * Implement /register, /deregister, and /discover endpoints

    * Manage heartbeats and TTLs

  - Technologies: FastAPI/Flask or Node.js (Express)

  - Challenges: Reliable discovery and registration

API Contract (v1.0) - Global Standards:

1. REST/HTTP communication using JSON payloads.

2. Authentication via shared secret in header.

3. Manager generates unique task_id (UUID) per user request.

4. Logging for all services to Overseer.

5. Services discover each other dynamically using Directory.

6. Standardized error response format (JSON).

Unit Testing Mandate ("Code the Contract"):

* Use a lightweight mock server (FastAPI/Flask/Express/Postman Mocks).

* Test in isolation for success and failure cases.

* Services must dynamically fetch discovered URLs via Directory.

Judging Criteria:

* Functionality, adherence to API contract, stability, error handling, observability, and compliance

accuracy.

Suggested Technologies:

Python (FastAPI/Flask), Node.js (Express), LangChain, Redis, ChromaDB, FAISS, Socket.IO, Postman, Mock Server.

End of Document.