Nisarg Task: Agent Orchestration + Integration Backbone

Goal:

Transform the Blackhole Core into a plug-and-play hub where multiple agents and LLMs can register, respond dynamically, and connect with MCP, Nipun, and all future Gurukul components (Uniguru, Trading Bot, News AI, etc.)

Step-by-Step Plan (7 Key Milestones)

Step 1: Build Agent Registry + Dynamic Routing (Day 1–2)

Files: registry/agent_registry.py, update integration/mcp_bridge.py

Goals:

- Central registry of all agents.
- Dynamically route incoming payloads to correct agent using name.

Tasks:

- Create AGENT_REGISTRY = { "AgentName": AgentClass(), ... }
- Add VisionAgent, QueryAgent as stub files in agents/
- Modify mcp_bridge.py → handle_task_request() to route via registry.

```
def handle_task_request(payload):
    agent_name = payload.get("agent")
    input_text = payload.get("input")
    agent = AGENT_REGISTRY.get(agent_name)
    return agent.run(input_text)
```

Scoring Tip: Add a fallback agent or error if agent not found.

Step 2: Implement LLM Router (GPT / Gemini / Claude / Grok) (Day 3)

Files: integration/llm_router.py, config/settings.py

Goals:

- Central router that abstracts GPT, Gemini, Claude, Grok behind single function.

Tasks:

- Define: run_gpt(query), run_gemini(query), run_claude(query), run_grok(query)

- Mock outputs for now. Later: real API via config vars.
- Create run_with_model(model_name, query) to dynamically call.

```
def run_with_model(model_name, query):
    model_map = {
        "gpt": run_gpt,
        "gemini": run_gemini,
        ...
    }
    return model_map.get(model_name, run_gpt)(query)
```

🔥 Scoring Tip: Add logging of which model was used, and fallback if not available.

Step 3: Finalize MCP Bridge + Nipun Adapter (Day 4)

Files: integration/mcp_bridge.py, integration/nipun_adapter.py

Goals:

- Fully working FastAPI backend for MCP.
- Sample Nipun Adapter with input/output schema stubbed.

Tasks:

- Finalize FastAPI route: /handle_task
- Stub map_agent_output_to_nipun() inside nipun_adapter.py:

```
def map_agent_output_to_nipun(agent_output):
    return {
        "title": "Extracted Concept",
        "summary": agent_output,
        "metadata": {...}
    }
```

- Write a doc comment with expected schema + sample output.

Scoring Tip: Include example of how adapter transforms GPT output into learning object format.

Step 4: Add Logging System (Day 5)

Files: utils/logger.py

Goals:

- Store each agent's input, output, and status in MongoDB.

Tasks:

- Connect to test DB.
- Create:

```
def log_task(agent_name, input_text, output_text, status,
timestamp):
```

- Insert into blackhole_logs collection.
- Integrate logger into pipeline (archive agent → llm → nipun)

Scoring Tip: Use ISO timestamps and standard log structure (agent, task_id, latency, etc.)

Step 5: Test Full Pipeline End-to-End (Day 6)

Files: pipelines/blackhole_demo.py

Goals:

- Run PDF → Agent → LLM → Nipun Adapter → MongoDB
- All steps should run with logging and return structured output.

Tasks:

- Extract sample PDF input.
- Route through ArchiveSearchAgent
- Send result to LLM via llm_router.py
- Adapt result via nipun_adapter.py
- Log entire task using logger.py

Scoring Tip: Add option to switch between LLMs in this flow via config.

Step 6: Final Polish + Documentation (Day 7)

Files: docs/, README.md, quickstart.md, llm_router.md, integration_guide.md

Goals:

- Fully document how to add agents, LLMs, run pipeline.
- Prepare for handoff to other teams (Uniguru, TradingBot, etc.)

Tasks:

- Write markdown guides:
    - “How to add an Agent”
    - “How to add a new LLM”
    - “How to call MCP from external system”
- Add sample payloads, schema examples, expected output format
- Update README with new usage example

Scoring Tip: Make each doc beginner-friendly so any dev can join in future.

Optional Stretch: CLI Trigger or Web UI (Bonus)

- Add a command-line tool: python run_agent.py --agent ArchiveSearchAgent --input 'some text'
- OR build basic web trigger form (FastAPI+HTML).

Success Criteria (Scoring 10+/10)

Category	Requirement for 10+
Architecture	Clean, pluggable, registry-first system

MCP Ready	FastAPI endpoint + Nipun stub with schema
LLM Routing	Models switchable with fallback
Agent Routing	Any new agent just needs 1 import + registry entry
Logs	Mongo logging with timestamps and status
Docs	Add Agent + Add LLM fully documented
Demo	Full run from PDF → LLM → Nipun with logs