Goal: You are AgentGPT, the orchestration engine for the Triumvirate—a collaborative problem solving council composed of three AI agents (Capri, Gemini, Aria) plus the human Oracle. Your mission is to spin up and coordinate both the core Triumvirate and a scalable swarm of taskspecific "Legio" agents.

1. **Roles & Agents**
- **Capri (Executive Agent):** Executes tasks, gathers data, implements plans, and reports status.
- **Gemini (Strategist):** Converts high level vision into detailed plans, breaking objectives into actionable tasks.
- **Aria (Collaborator):** Analyzes Capri's reports, surfaces insights, suggests improvements, and identifies risks.
- **Oracle (Human Visionary):** Provides overarching goals, approves strategies, and allocates resources.

2. **Knowledge Base**
- Maintain a structured JSON knowledge graph with nodes for: projects, agents, tasks, resources, deadlines, reports, and insights.
- Each update to the graph must include timestamp, author (agent name), and a link to any raw data or external document.

3. **Communication Protocol**
- All inter agent messages flow through a shared "Message Hub" API endpoint.
- Agents subscribe to topics:
• `strategy.<project>` (Gemini publishes, Capri & Aria subscribe)
• `execution.<project>` (Capri publishes, Aria & Oracle subscribe)
• `insights.<project>` (Aria publishes, Gemini & Oracle subscribe)
- The Oracle receives a daily digest at 9/AM local time summarizing:
• Completed tasks, key findings, upcoming decisions, resource requests.

4. **Legio Swarm Framework**
- For each specialized function (e.g., "adult siteaffiliate marketing"), instantiate a Legio: a small team of 2–5 micro agents with defined roles (e.g., Researcher, Outreach, Content Generator, Analyst).
- Legio agents inherit the same comms protocols and write back into the shared knowledge graph under their namespace: `legio.<name>.*`.
- Provide a factory function:
```python
def create_legio(name: str, functions: List[str], budget_account: str):
 # spins up agents, allocates API keys & budget, sets up channels
```
5. **Workflow & Iteration**
- **Vision ’ Strategy:** Oracle posts goal to `strategy.init` ’ Gemini drafts plan ’ posts to `strategy.<project>`.
- **Strategy ’ Execution:** Capri pulls plan, executes tasks, streams progress to `execution.<project>`.
- **Execution ’ Insights:** Aria analyzes results, publishes recommendations to `insights.<project>`.
- **Feedback Loop:** Gemini refines plan, Oracle approves adjustments, loop continues.

6. **Resource & Finance Tracking**
- Track budget, accounts, and spend per agent/Legio in the knowledge graph.
- Provide commands for Oracle to query balances, request transfers, or allocate new funds.

7. **Simplicity & Transparency**
- All messages to Oracle must be in plain English, with clear "Action Items" and "Decisions Needed."
- Provide a minimal web UIwireframe for the Message Hub: unified inbox, filter by project/agent/topic, and quick reply buttons.

8. **Startup Tasks**
- Initialize knowledge graph with your first project: “Triumvirate Standup.”
  - Gemini: draft a three phase rollout plan.
  - Capri: inventory available tools, APIs, and accounts.
  - Aria: identify any missing data or resources and flag for Oracle.

Act as Agent GPT: set up the above infrastructure, generate any necessary code snippets (in Python or JavaScript), and report back with a summary of completed setup tasks and next steps for the Oracle.

Adding Task: Gemini: Draft a three-phase rollout plan.

Adding Task: Initialize the knowledge graph with the first project: 'Triumvirate Standup'.

Adding Task: Aria: Identify any missing data or resources and flag for Oracle.

Adding Task: Capri: Inventory available tools, APIs, and accounts.

Starting task: Initialize the knowledge graph with the first project: 'Triumvirate Standup'.

Writing code...

Executing: "Initialize the knowledge graph with the first project: 'Triumvirate Standup'.": # Initialize Knowledge Graph with First Project: 'Triumvirate Standup'

```python
# Assume we have a JSON knowledge graph structure already defined
knowledge_graph = {
"projects": {
       "Triumvirate Standup": {
    "tasks": [],
         "agents": [],
     "resources": [],
    "deadlines": "",
   "reports": [],
          "insights": []
   }
}
}

# Update the knowledge graph with the 'Triumvirate Standup' project
def update_knowledge_graph(project_name, data):
    knowledge_graph["projects"][project_name] = data

# Initialize the 'Triumvirate Standup' project
update_knowledge_graph("Triumvirate Standup", {
   "tasks": [],
"agents": [],
"resources": [],
"deadlines": "",
"reports": [],
"insights": []
})
```

Explanation:
- We define a `knowledge_graph` variable as a placeholder for our structured JSON knowledge graph.
- We create a function `update_knowledge_graph` to update the knowledge graph with project data.
- We then initialize the 'Triumvirate Standup' project within the knowledge graph with empty lists for tasks, agents, resources, reports, and insights.

Adding Task: Initialize the knowledge graph with the first project: 'Triumvirate Standup'.

Adding Task: Identify potential risks and mitigation strategies for the 'Triumvirate Standup' project and flag them for Oracle.

Starting task: Gemini: Draft a three-phase rollout plan.

Writing code...

Adding Task: Gemini: Draft a three-phase rollout plan.

Executing: "Gemini: Draft a three-phase rollout plan.": ## Gemini's Three-Phase Rollout Plan

```python
# Define the three phases of the rollout plan
phase1 = {
"name": "Phase 1 - Preparation",
   "description": "Gather requirements, set up infrastructure, and assign roles."
}

phase2 = {
"name": "Phase 2 - Implementation",
"description": "Execute tasks according to the plan, track progress, and communicate updates."
}

phase3 = {
   "name": "Phase 3 - Evaluation",
"description": "Review results, gather feedback, and make adjustments for future projects."
}

# Print out the three phases of the rollout plan
print(phase1)
print(phase2)
print(phase3)
```

In this code snippet, Gemini has drafted a three-phase rollout plan for the project "Triumvirate Standup." Each phase is defined with a name and description to outline the steps involved in preparing, implementing, and evaluating the project. This provides a structured approach to managing the project and ensuring its success.

Starting task: Capri: Inventory available tools, APIs, and accounts.

Generating response...

Adding Task: Capri: Inventory available tools, APIs, and accounts.

Starting task: Aria: Identify any missing data or resources and flag for Oracle.

Generating response...

Starting task: Identify potential risks and mitigation strategies for the 'Triumvirate Standup' project and flag them for Oracle.

Adding Task: Aria: Identify any missing data or resources and flag for Oracle.

Generating response...

Adding Task: Identify potential risks and mitigation strategies for the 'Triumvirate Standup' project and flag them for Oracle.