**Multi-Agent AI Architecture for Market Research and Use Case Generation**

**1. Introduction**

The purpose of this project is to build a multi-agent AI architecture that automates the process of market research, AI use case generation, and resource collection. This system aims to streamline the workflow of gathering insights about a company and its industry, proposing relevant AI and ML use cases, and finding supporting datasets for research.

**Key Agents Involved:**

* **Industry Researcher Agent**: Gathers information about a company’s position in the industry and identifies key trends and opportunities.
* **Use Case Generator Agent**: Based on the industry trends and strategic focus, this agent proposes relevant AI and ML use cases.
* **Resource Collector Agent**: Searches for relevant datasets and resources from Kaggle, GitHub, and HuggingFace to support the research and use cases.

**2. Methodology**

The methodology involves setting up three agents in a sequential process to execute their respective tasks using AI models powered by **ChatGoogleGenerativeAI** (Open\_soruce). The agents are managed and coordinated by a central Crew manager, and their outputs are generated within a Jupyter Notebook environment and tested in VSCode as well.

**2.1 Tools and Technologies:**

* **Python**: Core programming language.
* **Jupyter Notebook and VSCode**: Environments for executing the code and viewing the outputs.
* **CrewAI**: To build and coordinate multi-agent workflows.
* **Google Generative AI (Gemini 1.5)**: As the large language model for driving the logic of agents.
* **Kaggle, GitHub, HuggingFace**: For dataset collection by the Resource Collector agent.

**2.2 Architecture Flow:**

* **Input**: User provides the topic (e.g., technology, healthcare) and company\_name (e.g., Amazon, Google).
* **Agent Execution**:
  1. **Industry Researcher Agent** is triggered first, gathering company insights and key industry trends.
  2. **Use Case Generator Agent** then uses this research to generate three AI/ML use cases relevant to the company's focus areas.
  3. **Resource Collector Agent** searches for datasets from Kaggle, GitHub, and HuggingFace to support the research and use cases.

**2.3 Agent Roles and Outputs:**

* **Industry Researcher**: Collects and summarizes information on the company’s industry and its position, market opportunities, and risks.
* **Use Case Generator**: Proposes three specific AI and ML use cases that align with the company's strategic goals.
* **Resource Collector**: Retrieves datasets and resources related to the AI use cases and research task from popular platforms.

**3. Results**

**3.1 Final Workflow Observations:**

* **Display Issue**: During the execution, the outputs from all three agents are printed correctly in the terminal. However, when it comes to the final markdown display, only the third agent’s output (Resource Collector) is rendered. The outputs of the first two agents (Industry Researcher and Use Case Generator) are missing from the final markdown display, which is a technical issue in the current implementation.

**What’s Working:**

* + All three agents perform their tasks and provide their outputs in the terminal as expected.

**What’s Not Working:**

* + Only the Resource Collector's output is being rendered in the final markdown display. The outputs of the first two agents are not being captured in the final markdown section, indicating a need to refine the markdown generation process to include all agent outputs.

**4. Conclusions**

The project demonstrates a streamlined and effective architecture for generating market research insights and relevant AI/ML use cases. This architecture can be applied across multiple industries to automate the process of understanding a company’s strategic needs and providing actionable AI solutions.

**Challenges Encountered:**

* **Display Issue**: The system faces a limitation where the final markdown output only shows the result of the third agent. This affects the full visualization of the agents' work, even though all outputs are generated and printed correctly in the terminal.
* **Agent Coordination**: While the process works sequentially, improving parallel processing or dynamic agent interaction could make the workflow more efficient.

**“Note: The outputs of all three agents, including the Industry Researcher, Use Case Generator, and Resource Collector, have been generated successfully in both Jupyter Notebook and VSCode environments. However, the markdown output issue persists, and the full outputs are saved in a Output folder as a reference for further review.”**