Multi-Agent System Documentation

# 1. Introduction

This document provides an overview of a Multi-Agent System (MAS) designed for AI-driven solutions for businesses. The system uses multiple agents to perform tasks like analyzing industries, generating use cases for AI/ML applications, and suggesting GenAI solutions. The system is built around three core agents: IndustryAgent, UseCaseAgent, and GenAISolutionsAgent, which collaborate to offer AI-powered insights to businesses.

# 2. System Overview

The Multi-Agent System is designed to help businesses in various industries by providing AI-driven solutions. The system employs three distinct agents, each responsible for specific tasks, which collectively provide comprehensive insights.

## 2.1 Core Agents

The system includes the following core agents:  
- IndustryAgent: Analyzes industry-specific information for a given company.  
- UseCaseAgent: Generates AI/ML use cases based on the provided industry information.  
- GenAISolutionsAgent: Suggests GenAI-based solutions (e.g., automated reporting, chat systems) based on the generated use cases.

# 3. Agent Details

## 3.1 IndustryAgent

The IndustryAgent is responsible for analyzing the industry for a specific company. Using the Gemini API, it generates a brief overview of the industry, focusing on key trends, opportunities, and strategic areas relevant to the company.

Key Functionality:  
- `analyze\_industry(company\_name)`: Generates a prompt to provide a brief overview of the industry for the specified company.

## 3.2 UseCaseAgent

The UseCaseAgent generates AI/ML use cases based on the industry information provided. It structures each use case by describing the objective, AI application, and cross-functional benefits for different departments.

Key Functionality:  
- `generate\_use\_cases(industry\_info)`: Generates AI/ML use cases based on the provided industry information.

## 3.3 GenAISolutionsAgent

The GenAISolutionsAgent suggests relevant GenAI solutions based on the generated AI/ML use cases. These solutions could include applications such as document search, automated reporting, and chat systems.

Key Functionality:  
- `suggest\_genai\_solutions(use\_cases)`: Proposes GenAI-based solutions for the generated use cases.

# 4. Multi-Agent Orchestrator

The Multi-Agent Orchestrator serves as the central controller for managing the interaction between the agents. It coordinates the tasks of the IndustryAgent, UseCaseAgent, and GenAISolutionsAgent to provide a seamless workflow.

## 4.1 Workflow Steps

The orchestrator follows these steps:  
1. Collects industry information using the IndustryAgent.  
2. Generates AI/ML use cases based on the industry information using the UseCaseAgent.  
3. Suggests GenAI solutions based on the generated use cases using the GenAISolutionsAgent.

## 4.2 Methods in Multi-Agent Orchestrator

The following methods are used in the Multi-Agent Orchestrator:  
- `collect\_industry\_info()`: Collects industry information for the given company.  
- `collect\_use\_cases()`: Generates AI/ML use cases based on industry information.  
- `collect\_genai\_solutions()`: Suggests GenAI solutions based on generated use cases.  
- `run\_workflow()`: Executes the entire workflow, collecting industry information, use cases, and GenAI solutions.

# 5. Example Usage

To use the Multi-Agent System, simply provide the company name and run the workflow. The system will return the generated industry information, use cases, and suggested GenAI solutions.

## 5.1 Example Code

# Initialize the Orchestrator with a company name  
 orchestrator = MultiAgentOrchestrator(company\_name="Example Company")  
  
 # Run the workflow  
 results = orchestrator.run\_workflow()  
  
 # Print the results  
 print(results)

# 6. Conclusion

This Multi-Agent System demonstrates a powerful framework for analyzing industries, generating AI/ML use cases, and suggesting GenAI solutions. The modular approach using different agents allows flexibility and scalability to adapt to different business needs.