Thank you for the clarification. I understand that our focus should be on multi-agent interactions, specifically exploring the possibility of using multiple LLMs simultaneously. Let's update our blueprint to reflect this focus.

[Tool Use: write\_to\_file] Path: docs/CAMEL\_Blueprint.md Content: # CAMEL Adaptation Blueprint

**Overview**

This document outlines the blueprint for adapting the CAMEL project to meet the specific needs of our ADAPT-CamelDEV implementation. Our primary focus is on multi-agent interactions, particularly exploring the possibility of using multiple LLMs simultaneously.

**Key Concepts**

* **LLM-Based Multi-Agent Framework:** CAMEL's core strength lies in its use of large language models (LLMs) for building multi-agent systems.
* **Multiple LLM Integration:** We aim to extend CAMEL's capabilities to support simultaneous use of multiple LLMs, including our GitHub and Gemini models.

**Multi-Agent Interactions with Multiple LLMs**

1. **Simultaneous LLM Usage:**
   * Implement a system where multiple LLMs (e.g., GitHub model, Gemini model) can be active and collaborating simultaneously.
   * Develop a coordination mechanism to manage inputs and outputs between different LLMs.
2. **Specialized Agent Roles:**
   * Assign different LLMs to specialized roles based on their strengths:
     + GitHub Model: Code analysis, repository management, issue tracking
     + Gemini Model: Natural language processing, general knowledge tasks
   * Create a flexible role assignment system that can dynamically allocate tasks to the most suitable LLM.
3. **Inter-LLM Communication:**
   * Design a standardized message format that allows different LLMs to exchange information effectively.
   * Implement translation layers if necessary to ensure compatibility between different LLM outputs.
4. **Task Distribution and Aggregation:**
   * Develop a system to break down complex tasks into subtasks that can be distributed among multiple LLMs.
   * Create an aggregation mechanism to combine outputs from different LLMs into a cohesive result.
5. **Parallel Processing:**
   * Implement parallel processing capabilities to allow multiple LLMs to work on different aspects of a task simultaneously.
   * Develop synchronization mechanisms to manage dependencies between parallel LLM operations.
6. **Conflict Resolution:**
   * Design a system to detect and resolve conflicts that may arise from contradictory outputs of different LLMs.
   * Implement a voting or confidence-scoring mechanism to decide between conflicting results.

**Implementation Plan**

1. **LLM Integration:**
   * Adapt the ModelFactory to support simultaneous initialization and management of multiple LLM models.
   * Implement interfaces for GitHub and Gemini models to ensure consistent interaction within the CAMEL framework.
2. **Multi-Agent Coordinator:**
   * Develop a high-level coordinator agent that orchestrates interactions between multiple LLMs.
   * Implement task distribution and result aggregation logic in the coordinator.
3. **Message Handling:**
   * Extend the BaseMessage class to support routing between different LLMs.
   * Implement message translation or adaptation layers if required for inter-LLM communication.
4. **Parallel Processing Framework:**
   * Develop a parallel execution engine that can manage multiple LLM operations concurrently.
   * Implement task dependency management and synchronization mechanisms.
5. **Conflict Resolution Module:**
   * Create a module to detect and manage conflicts between LLM outputs.
   * Implement resolution strategies (e.g., voting, confidence scoring, human intervention).
6. **Testing and Optimization:**
   * Develop comprehensive test scenarios to validate multi-LLM interactions.
   * Optimize resource allocation and load balancing for efficient multi-LLM operations.

**Next Steps**

1. Implement basic multi-LLM support in the ModelFactory.
2. Develop a prototype of the multi-agent coordinator.
3. Create test scenarios that demonstrate simultaneous use of GitHub and Gemini models.
4. Implement and test the parallel processing framework.
5. Develop the conflict resolution module and integration tests.
6. Continuously refine and optimize the multi-LLM interaction capabilities.

This blueprint will be updated as we implement and test the multi-LLM capabilities, incorporating new insights and optimizing our approach to multi-agent interactions.

**API Request**

$0.3754

Now that we have updated our CAMEL\_Blueprint.md with a focus on multi-agent interactions and the simultaneous use of multiple LLMs, we should proceed to the next step in our task. Let's explore the existing codebase to see if there are any components we can leverage or adapt for our multi-LLM implementation.

[Tool Use: list\_files] Path: camel Recursive: false