Building Multi-Agent Applications with Semantic Kernel: A Detailed Plan and Conceptual Code Examples

Introduction: The Rise of Multi-Agent Systems and Semantic Kernel.

The complexity of modern problems often necessitates solutions that go beyond the capabilities of single, monolithic systems. Multi-agent systems, comprising several autonomous entities that interact to achieve a common objective, have emerged as a powerful paradigm for tackling such intricate challenges. These systems distribute intelligence and workload, allowing for specialization and collaboration that can lead to more robust and efficient outcomes. In this evolving landscape of artificial intelligence, Microsoft's Semantic Kernel stands out as a versatile framework designed to facilitate the creation of intelligent, collaborative AI applications.[1]

Semantic Kernel provides developers with the tools to integrate large language models (LLMs) with conventional programming languages like C#, Python, and Java.[1] Its architecture is built around key features that are particularly relevant to multi-agent systems, including the orchestration of plugins (functions), the ability to create and execute plans, and sophisticated memory management capabilities.[1] The growing adoption of multi-agent systems reflects a significant trend in AI, moving towards distributed problem-solving where specialized agents can collaborate to accomplish goals that would be unattainable for a single agent. This mirrors the dynamics of human teams, where diverse skills and perspectives are brought together to achieve a common objective. Semantic Kernel's design directly addresses this trend by offering a structured environment for building and managing such collaborative AI systems. Furthermore, the framework's model-agnostic nature is a critical advantage, allowing for the seamless integration of various LLMs and AI services from providers like OpenAI, Azure OpenAI, and Hugging Face.[1] This flexibility enables the construction of multi-agent systems where different agents can leverage the unique strengths of different underlying AI models, optimizing the overall performance and capabilities of the application.

Understanding Core Concepts of Multi-Agent Systems within Semantic Kernel.

The Semantic Kernel Agent Framework provides the necessary abstractions for building multi-agent systems. At the core of this framework is the concept of an Agent, which represents an autonomous or semi-autonomous software entity designed to perform tasks by processing inputs, making decisions, and producing outputs.[11] These agents can engage in conversations, sending and receiving messages, and their responses are generated through a combination of models, tools

(plugins), human input, and other customizable components.[11] The Agent Framework simplifies the creation of specialized AI entities, each possessing distinct personas and access to specific tools, thereby forming the fundamental building blocks of a multi-agent system.[15] Different types of agents are available within the framework, including ChatCompletionAgent, which leverages Semantic Kernel's chat completion capabilities, and OpenAIAssistantAgent, which utilizes the OpenAI Assistant API for more advanced functionalities.[15] Developers can also create custom agents by extending the base Agent class to implement unique behaviors and interaction patterns.[15] Each agent within the framework relies on a Kernel instance, which provides access to essential services and plugins necessary for the agent to function.[15]

Beyond the creation of individual agents, establishing effective interaction patterns between them is crucial for achieving the common goal. Semantic Kernel provides the AgentChat abstraction to manage conversations involving multiple agents.[14] This framework supports various coordination strategies, including group chats where multiple agents and humans can participate and share a common chat history, as well as more structured task-based business process flows.[14] For collaborative scenarios requiring defined interaction rules, AgentGroupChat can be utilized, allowing for the specification of selection strategies (determining which agent speaks next) and termination strategies (defining when the conversation or task is complete).[16] Semantic Kernel's provision of these specific mechanisms for managing the flow of interactions and collaboration among multiple agents addresses the inherent complexity of coordinating autonomous entities, ensuring that they can work together effectively to solve complex problems.

Exploring Diverse Agent Types in Semantic Kernel for Collaborative Tasks.

When designing a multi-agent system, it is beneficial to consider the different ways in which agents can be categorized based on their roles and responsibilities. One key distinction is between specialized agents and task-specific agents. Specialized agents are designed with specific skills and knowledge domains, making them efficient at handling particular types of tasks. Examples of specialized agents could include an agent focused on data scraping from websites, another dedicated to financial analysis, or an agent skilled in generating travel plans.[12] These agents are akin to experts in their respective fields, bringing deep knowledge and specific capabilities to the collaborative effort. Conversely, task-specific agents are primarily focused on executing particular steps within a larger workflow. These agents might be responsible for a single action or a small set of related actions that contribute to the overall objective.[12] For instance, in a multi-step process, one agent might be

responsible for retrieving data, while another processes that data, and a third generates a report. The design of a multi-agent system benefits significantly from this categorization, as it leads to a more efficient and modular architecture where each agent's purpose is clearly defined.

Selecting the appropriate agent types for a multi-agent application requires a thorough analysis of the common goal and the identification of the necessary agent capabilities.[24] The types of agents chosen should directly align with the specific tasks or roles that need to be performed within the overall workflow.[24] For example, if the goal involves gathering information from the web, an agent with web scraping capabilities would be essential. Similarly, if the system needs to analyze financial data, a financial analysis agent would be required. It is also important to consider the trade-offs between using highly specialized agents, which excel in narrow domains, and more general-purpose agents, which might be capable of handling a broader range of tasks but perhaps with less expertise in any single area. Ultimately, careful consideration of the roles and responsibilities required to achieve the common goal is crucial for selecting the right types of agents and ensuring effective collaboration within the multi-agent system.

Leveraging Semantic Kernel Planners for Orchestrating Multi-Agent Workflows.

Semantic Kernel provides a powerful set of planners that are instrumental in orchestrating the interactions and workflows between multiple agents in a coordinated manner. These planners enable the system to automatically determine the sequence of actions and agent interactions required to achieve a common goal.[23] While the Sequential Planner, which executed plugins in a predefined order, passing outputs between steps, is no longer supported and Handlebars Planner is the recommended alternative [18], other planners like the Function Calling Stepwise Planner and the Handlebars Planner offer robust capabilities for multi-agent orchestration.

The Function Calling Stepwise Planner allows the AI to form "thoughts" and "observations" and execute actions step-by-step until the user's goal is achieved.[30] This planner can dynamically select and invoke different plugins (and thus different agents) based on the current state and the overall objective. It continues to iterate through a process of thinking, observing, and acting until all required functions are complete and a final output is generated. This dynamic nature makes the Stepwise Planner well-suited for complex tasks with interconnected steps where the optimal sequence of actions might not be known in advance.

The Handlebars Planner utilizes the Handlebars templating language to generate the

plan.[31] This approach offers the advantage of leveraging a widely supported templating language, which can improve the accuracy of the generated plan as many LLMs already support the Handlebars interface when dealing with prompts. The Handlebars Planner can also incorporate loops and conditional logic into the plan, providing a high degree of flexibility in defining multi-agent workflows. Furthermore, the plan generated by the Handlebars Planner is often in a readable text format that can be stored and reloaded, which can be beneficial for auditing and reusing plans. Semantic Kernel's planners, particularly the Stepwise and Handlebars Planners, are therefore essential tools for managing the complexity of multi-agent workflows by automating the decision-making process of which agent to invoke and in what order to achieve the desired outcome.

Defining Clear Roles and Responsibilities for Agents in a Collaborative Ecosystem.

For a multi-agent system to function effectively, it is crucial to define clear roles and responsibilities for each agent involved. This ensures that each agent understands its specific purpose and contribution to the overarching goal, ultimately leading to more efficient task completion and seamless collaboration.[13] When designing a multi-agent application with 6-7 distinct agents, each agent should be assigned a specific role based on the chosen use case. For example, in a collaborative project management tool, potential roles could include a Project Manager responsible for defining tasks and monitoring progress, a Task Planner focused on breaking down projects into actionable steps, a Developer tasked with executing coding assignments, a Tester responsible for quality assurance, a Documenter for creating project documentation, and a Client Liaison for communication with stakeholders. Optionally, a Risk Assessor could be included to identify and analyze potential project risks.

Defining the specific responsibilities and tasks associated with each role is equally important. The Project Manager, for instance, would be responsible for outlining project objectives, setting deadlines, and assigning tasks to the appropriate agents. The Task Planner would utilize AI planning capabilities to generate a detailed project plan with individual steps. The Developer would focus on writing and implementing the code required for their assigned tasks. The Tester would design and execute test cases to ensure the quality of the developed features. The Documenter would create user manuals, technical specifications, and other necessary documentation. The Client Liaison would gather requirements from the client, provide regular updates on the project's progress, and address any concerns. Finally, the Risk Assessor would analyze project parameters and external factors to proactively identify and communicate potential risks to the Project Manager. It is essential to ensure that the

responsibilities of each agent are clearly delineated and non-overlapping to avoid confusion, conflicts, and redundant efforts within the multi-agent system. While defining these static roles is a fundamental step, the ability for agents to adapt their roles or for the system to dynamically reassign tasks based on the current state can further enhance the robustness and efficiency of the application.[24]

Establishing Inter-Agent Communication Channels in Semantic Kernel.

Effective communication between agents is paramount in a multi-agent system to ensure coordinated action and information flow. Semantic Kernel provides several mechanisms to facilitate this inter-agent communication, including context sharing, message passing, and the potential for utilizing a central message bus. One fundamental approach is through context sharing. The Kernel object in Semantic Kernel maintains a collection of ContextVariables, which can be used to share information between different agents.[45] When one agent executes a function, its output can be stored in the ContextVariables, making it accessible to other agents within the same workflow. Furthermore, within a planned workflow orchestrated by a Semantic Kernel planner, the output of one agent's function can be directly passed as input to another agent's function, creating a seamless chain of actions.[29] Shared memory, as discussed in a subsequent section, also serves as a crucial mechanism for agents to maintain common knowledge and exchange information. Semantic Kernel's inherent ability to pass context and results between functions (and consequently, agents) forms a cornerstone for enabling coordinated action and a smooth flow of information within a multi-agent system.

In addition to context sharing, Semantic Kernel's Agent Framework offers message passing techniques through the AgentChat abstraction.[14] This framework provides a dedicated channel for agents to communicate using messages, offering a more explicit and potentially more robust way to manage interactions compared to simply relying on shared context variables. Agents participating in an AgentChat can send and receive messages, with their input and output typically aligned to the ChatMessageContent type.[19] This structured approach to message exchange can be particularly beneficial for more complex multi-agent scenarios where clear and auditable communication logs are required. While not explicitly provided as a built-in component, the potential exists to implement a central message bus for more decoupled communication between agents. This could involve integrating an external messaging system, such as one based on a publish/subscribe pattern, with Semantic Kernel. A central message bus would allow agents to communicate without having direct dependencies on each other, potentially enhancing the scalability and maintainability of the multi-agent system, although it might introduce additional

complexity to the implementation.

Managing and Sharing Memory Among Multiple Agents for Contextual Awareness.

Memory management is a critical aspect of building effective multi-agent systems, as it allows agents to retain information from past interactions and maintain contextual awareness. Semantic Kernel offers various options for managing and sharing memory among multiple agents, including both short-term and long-term memory solutions. For short-term, session-specific memory, the VolatileMemoryStore can be utilized.[55] This in-memory store allows agents to retain data for the duration of a session or interaction, which can be useful for maintaining immediate context, such as the flow of a conversation or task-related data that does not need to be recalled later. For long-term memory needs, Semantic Kernel provides SemanticTextMemory, which can be configured with various storage backends, including Azure AI Search, Chroma, and Milvus.[58] This type of memory allows for the semantic indexing of information using embeddings, enabling agents to perform similarity searches and retrieve relevant knowledge based on meaning rather than just keywords. Furthermore, Kernel Memory represents a more comprehensive memory management service within the Semantic Kernel ecosystem.[56] It offers features beyond basic semantic memory, such as the ability to handle various data formats (including web pages, PDFs, and images), perform hybrid searches with filters, and manage document ingestion at scale.

Shared memory can serve as a central knowledge repository for the entire multi-agent system, enabling agents to collaboratively solve problems and maintain a consistent understanding of the task at hand. Strategies can be implemented to allow agents to access and update this shared memory, ensuring that intermediate results, decisions, and learnings from different agents are stored and can be utilized by others. For instance, one agent might perform an analysis and store the findings in shared memory, which can then be accessed by another agent to generate a report. When using shared memory, it is important to consider the need for access control mechanisms to ensure data privacy and security, as well as strategies for maintaining data consistency across multiple agents that might be reading and writing to the same memory store.

Best Practices for Structuring Multi-Agent Semantic Kernel Applications.

Structuring a multi-agent Semantic Kernel application effectively is crucial for ensuring modularity, maintainability, and scalability, especially when dealing with 6-7 or more agents. One recommended best practice is to separate the definitions

(instructions) of the agents from their functional code (plugins).[15] This can be achieved by organizing the project into logical directories, such as a Plugins/ directory to house the code for native and semantic plugins, and a Prompts/ directory to store the prompt templates and configuration files that define the behavior of semantic functions and agents.[21] Within these directories, further organization based on agent roles or specific functionalities can enhance maintainability. For example, plugins related to a specific agent or a particular aspect of the application could be grouped into subdirectories.

Utilizing dependency injection is another important best practice for managing agent dependencies and the various services they rely on.[75] Semantic Kernel's architecture is designed to support dependency injection, allowing developers to register services (such as AI connectors, memory stores, and custom utilities) with the Kernel and then inject them into plugins and agents as needed. This approach promotes loose coupling between components, making the application more flexible and easier to test and maintain. When designing the architecture for an application with 6-7 or more agents, it is important to choose an appropriate orchestration pattern. While a centralized planner can be effective for many scenarios, decentralized coordination models, where agents have more autonomy in deciding their actions, might be suitable for other use cases. It is also crucial to consider the potential performance implications of having a large number of interacting agents and to implement robust error handling and monitoring mechanisms.[80] Tools like Azure Monitor and Application Insights can be integrated to track agent performance, token usage, and other relevant metrics, providing valuable insights into the application's behavior and helping to identify and address any issues that arise.

Conceptual Code Examples Illustrating Multi-Agent Interactions in Semantic Kernel.

The following conceptual code examples illustrate the basic principles of creating and interacting with multiple agents in Semantic Kernel using Python. Similar patterns can be applied in C# and Java.

Python

```
# Python Example
```

```python
import semantic_kernel as sk
from semantic_kernel.agents import ChatCompletionAgent
from semantic_kernel.connectors.ai.open_ai import OpenAIChatCompletion

async def main():
    # Initialize the Kernel
    kernel = sk.Kernel()
    # Add your AI service connector (e.g., Azure OpenAI)
    kernel.add_service(OpenAIChatCompletion(service_id="my_llm",
model_id="gpt-3.5-turbo"))

    # Define Agent 1: Project Manager
    pm_instructions = "You are the project manager. Your goal is to oversee the project and ensure
tasks are completed on time."
    project_manager = ChatCompletionAgent(name="ProjectManager",
instructions=pm_instructions, kernel=kernel)

    # Define Agent 2: Developer
    dev_instructions = "You are a software developer. Your task is to write code according to the
project plan."
    developer = ChatCompletionAgent(name="Developer", instructions=dev_instructions,
kernel=kernel)

    # Define a simple task
    task = "Write a function that adds two numbers in Python."

    # Project Manager assigns the task to the Developer
    pm_response = await project_manager.get_response(messages=f"Developer, your task
is: {task}")
    print(f"{project_manager.name} says: {pm_response.content}")

    # Developer acknowledges the task
    dev_response = await developer.get_response(messages=f"Okay, ProjectManager. I will
start working on: {task}")
    print(f"{developer.name} says: {dev_response.content}")

if __name__ == "__main__":
    import asyncio
    asyncio.run(main())
```

This basic example demonstrates the creation of two agents with distinct roles and a simple interaction where the Project Manager assigns a task to the Developer. To incorporate a planner, we could extend this example to have the Project Manager use a planner to break down a larger project into multiple tasks and assign them to different agents.

Python

```
# Python Example with Planner (Conceptual)

import semantic_kernel as sk
from semantic_kernel.agents import ChatCompletionAgent
from semantic_kernel.connectors.ai.open_ai import OpenAIChatCompletion
from semantic_kernel.planners import SequentialPlanner # Or HandlebarsPlanner

async def main():
    # Initialize the Kernel
    kernel = sk.Kernel()
    kernel.add_service(OpenAIChatCompletion(service_id="my_llm",
model_id="gpt-3.5-turbo"))

    # Define agents (ProjectManager, Developer, Tester) as before

    # Add plugins (e.g., a TaskManagementPlugin) to the kernel
    # kernel.add_plugin(TaskManagementPlugin(), plugin_name="TaskManagement")

    # Initialize a planner
    planner = SequentialPlanner(kernel) # Or planner = HandlebarsPlanner(kernel)

    # Define the overall goal
    goal = "Develop and test a simple calculator application."

    # Create a plan
    plan = await planner.create_plan(goal=goal)

    print("Plan steps:")
```

```
    for step in plan._steps:
        print(f"- {step.description} using {step.metadata.fully_qualified_name}")

    # Execute the plan (conceptual - would involve invoking agents based on plan steps)
    # result = await plan.invoke(kernel)
    # print(f"Plan result: {result}")

if __name__ == "__main__":
    import asyncio
    asyncio.run(main())
```

These examples provide a foundational understanding of how to create agents and begin orchestrating their interactions using Semantic Kernel. The use of native and semantic plugins would further extend the capabilities of these agents, allowing them to perform specific actions and access external resources. Memory sharing could be implemented by storing and retrieving information from a SemanticTextMemory instance accessible to all relevant agents. Filters could be applied to control the input and output of agent interactions, ensuring adherence to specific guidelines or security protocols.

Concrete Use Case: A Multi-Agent Collaborative Project Management Tool.

Consider a multi-agent collaborative project management tool built using Semantic Kernel, featuring six distinct agents working together to manage a software development project.

1. **Project Manager:** This agent is responsible for initiating the project, defining the overall goals, setting milestones, and monitoring the progress of the other agents. It utilizes a planner to create the initial project plan and adjust it based on feedback and progress updates.
2. **Task Planner:** Working under the direction of the Project Manager, this agent takes the high-level project goals and breaks them down into a series of actionable tasks. It uses AI planning algorithms to optimize the task dependencies and assign estimated durations.
3. **Developer:** This agent is responsible for executing the coding tasks outlined in the project plan. It can access relevant code repositories, write and commit code, and communicate its progress to the Project Manager and Tester agents.
4. **Tester:** Once the Developer completes a task, this agent takes over to perform automated and manual testing. It can generate test reports, identify bugs, and communicate issues back to the Developer for resolution.

5. **Documenter:** This agent is responsible for creating and maintaining project documentation, including user manuals, technical specifications, and API documentation. It can automatically generate documentation based on the Developer's code and the Project Manager's specifications.
6. **Client Liaison:** This agent serves as the primary point of contact with the client. It gathers requirements, provides progress updates, and manages client feedback, relaying relevant information to the other agents as needed.

In this scenario, the Project Manager agent might begin by using a Handlebars Planner to outline the project's phases and key deliverables. The Task Planner agent would then take each deliverable and, using a Stepwise Planner, break it down into individual tasks, assigning them to the Developer and Tester agents based on their expertise and availability. The Developer agent would then execute its assigned coding tasks, updating the shared project context with its progress. Upon completion, the Tester agent would automatically initiate tests, logging any defects found in the shared context. The Documenter agent could monitor the Developer's code commits and, using semantic plugins, automatically generate corresponding documentation. Throughout this process, the Client Liaison agent would communicate with the client, providing updates derived from the shared project context and relaying any new requirements back to the Project Manager. Agents could communicate directly using AgentChat for specific queries or updates, while the shared project context in memory would ensure that all agents have a consistent view of the project's status.

Conclusion: Advancing Collaborative AI with Semantic Kernel.

In conclusion, Semantic Kernel provides a powerful and flexible framework for building sophisticated multi-agent applications. By offering abstractions for agents, various planning mechanisms, robust communication channels, and versatile memory management options, it empowers developers to create complex AI systems where multiple autonomous entities can collaborate effectively to achieve common goals. The ability to define specialized agents with clear roles and responsibilities, orchestrate their interactions using intelligent planners, facilitate seamless communication, and manage shared knowledge through memory are all key aspects that make Semantic Kernel a compelling choice for advancing the field of collaborative AI. As AI technology continues to evolve, frameworks like Semantic Kernel will play a crucial role in enabling the development of more intelligent, autonomous, and collaborative AI solutions capable of tackling increasingly complex real-world problems.

Works cited

1. microsoft/semantic-kernel: Integrate cutting-edge LLM technology quickly and easily into your apps - GitHub, accessed on April 10, 2025, https://github.com/microsoft/semantic-kernel/
2. microsoft/semantic-kernel: Integrate cutting-edge LLM technology quickly and easily into your apps - GitHub, accessed on April 10, 2025, https://github.com/microsoft/semantic-kernel
3. MicrosoftDocs/semantic-kernel-docs - GitHub, accessed on April 10, 2025, https://github.com/MicrosoftDocs/semantic-kernel-docs
4. Semantic Kernel | AI Hub - Azure documentation, accessed on April 10, 2025, https://azure.github.io/aihub/docs/concepts/semantic-kernel/
5. Semantic Kernel documentation | Microsoft Learn, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/
6. Introduction to Semantic Kernel | Microsoft Learn, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/overview/index
7. Add chat completion services to Semantic Kernel | Microsoft Learn, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/concepts/ai-services/chat-completion/
8. Semantic Kernel: The New Way to Create Artificial Intelligence Applications - Medium, accessed on April 10, 2025, https://medium.com/globant/semantic-kernel-the-new-way-to-create-artificial-intelligence-applications-7959d5fc90ca
9. Building AI agents with the Semantic Kernel SDK and Azure OpenAI | Will Velida, accessed on April 10, 2025, https://www.willvelida.com/posts/intro-to-semantic-kernel/
10. Chatbot with Semantic Kernel - Part 6: AI Connectors - DEV Community, accessed on April 10, 2025, https://dev.to/davidgsola/chatbot-with-semantic-kernel-part-6-ai-connectors-4b63
11. Semantic Kernel Agent Framework | Microsoft Learn, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/
12. Building Multi-Agent Systems with Multi-Models in Semantic Kernel - Part 1 - Arafat Tehsin, accessed on April 10, 2025, https://arafattehsin.com/building-multi-agent-systems-with-multi-models-in-semantic-kernel-part-1/
13. Multi-Agent Orchestration Redefined with Microsoft Semantic Kernel - Akira AI, accessed on April 10, 2025, https://www.akira.ai/blog/multi-agent-with-microsoft-semantic-kernel
14. Introducing enterprise multi-agent support in Semantic Kernel, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/introducing-agents-in-semantic-kernel/
15. Semantic Kernel Agent Architecture | Microsoft Learn, accessed on April 10, 2025,

https://learn.microsoft.com/en-us/semantic-kernel/frameworks/agent/agent-architecture
16. semantic-kernel/docs/decisions/0032-agents.md at main · microsoft ..., accessed on April 10, 2025, https://github.com/microsoft/semantic-kernel/blob/main/docs/decisions/0032-agents.md
17. Semantic Kernel Part 4: Agents - CODE Magazine, accessed on April 10, 2025, https://www.codemag.com/Article/2501061/Semantic-Kernel-Part-4-Agents
18. How to create a GenAI agent using Semantic Kernel | Nearform, accessed on April 10, 2025, https://www.nearform.com/digital-community/how-to-create-a-genai-agent-using-semantic-kernel/
19. Semantic Kernel Agents Overview | Restackio, accessed on April 10, 2025, https://www.restack.io/p/semantic-kernel-knowledge-answer-agents-cat-ai
20. Guest Blog: Step-by-Step Guide to Building a Portfolio Manager: A Multi-Agent System with Microsoft Semantic Kernel and Azure OpenAI, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/guest-blog-step-by-step-guide-to-building-a-portfolio-manager-a-multi-agent-system-with-microsoft-semantic-kernel-and-azure-openai/
21. Modularity and Abstraction in multi-agent applications | by Valentina ..., accessed on April 10, 2025, https://valentinaalto.medium.com/modularity-and-abstraction-in-multi-agent-applications-c566a510f142
22. Step-by-Step Guide to Building a Portfolio Manager: A Multi-Agent System with Microsoft Semantic Kernel and Azure OpenAI | by Akshay Kokane | AI Advances, accessed on April 10, 2025, https://ai.gopubby.com/step-by-step-guide-to-building-a-portfolio-manager-a-multi-agent-system-with-microsoft-semantic-639b7c899da2
23. Guest Blog: Building Multi-Agent Systems with Multi-Models in Semantic Kernel – Part 1, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/guest-blog-building-multi-agent-systems-with-multi-models-in-semantic-kernel-part-1/
24. How do multi-agent systems use role assignment? - Zilliz Vector Database, accessed on April 10, 2025, https://zilliz.com/ai-faq/how-do-multiagent-systems-use-role-assignment
25. What are multi-agent systems? - SAP, accessed on April 10, 2025, https://www.sap.com/swiss/resources/what-are-multi-agent-systems
26. Build a multi-agent system | Cognizant, accessed on April 10, 2025, https://www.cognizant.com/us/en/cmp/build-a-multi-agent-system
27. Multi-Agent Systems Reshape Human-Machine Collaboration, accessed on April 10, 2025, https://www.insurancethoughtleadership.com/ai-machine-learning/multi-agent-systems-reshape-human-machine-collaboration
28. Orchestrating Multi-Agent AI With Semantic Kernel | Digital Bricks, accessed on

April 10, 2025, https://www.digitalbricks.ai/blog-posts/orchestrating-multi-agent-ai-with-semantic-kernel
29. Introduction to Semantic Kernel Planners for Seamless Orchestration | by Akshay Kokane, accessed on April 10, 2025, https://medium.com/@akshaykokane09/empowering-ai-with-semantic-kernel-planners-for-seamless-orchestration-1c7ad35f2337
30. semantic-kernel/python/samples/getting_started/05-using-the-planner.ipynb at main, accessed on April 10, 2025, https://github.com/microsoft/semantic-kernel/blob/main/python/samples/getting_started/05-using-the-planner.ipynb
31. How to create a GenAI agent using Semantic Kernel | Nearform, accessed on April 10, 2025, https://nearform.com/digital-community/how-to-create-a-genai-agent-using-semantic-kernel/
32. Semantic Kernel Planner 101 - baeke.info, accessed on April 10, 2025, https://blog.baeke.info/2023/06/01/semantic-kernel-planner-101/
33. Using Planners in the SK Java Kernel | Semantic Kernel - Microsoft Developer Blogs, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/using-planners-in-the-sk-java-kernel/
34. Semantic Kernel Planner 101 - baeke.info, accessed on April 10, 2025, https://baeke.info/2023/06/01/semantic-kernel-planner-101/
35. Intro to Semantic Kernel – Part Two - Coding, accessed on April 10, 2025, https://blog.brakmic.com/intro-to-semantic-kernel-part-two/
36. Semantic Kernel - Sequential Planner - Microsoft Developer Blogs, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/semantic-kernel-planners-sequential-planner/
37. Getting Started with Semantic Kernel and C# - Matt on ML.NET - Accessible AI, accessed on April 10, 2025, https://accessibleai.dev/post/introtosemantickernel/
38. Diving into Microsoft Semantic Kernel | by Prarthana Saikia - Medium, accessed on April 10, 2025, https://medium.com/@prarthana1/diving-into-microsoft-semantic-kernel-82e037fe427c
39. How to create a sequential planner using semantic kernel - YouTube, accessed on April 10, 2025, https://www.youtube.com/watch?v=r-atDSeqLaI
40. Using intelligent planners in our AI Agents with the Semantic Kernel SDK and C# - YouTube, accessed on April 10, 2025, https://www.youtube.com/watch?v=yAZxPPc13GI
41. Semantic Kernel Hello World Planners Part 1 - Jason Haley, accessed on April 10, 2025, https://jasonhaley.com/2024/05/19/semantic-kernel-hello-world-planners-part1/
42. Semantic Kernel Hello World Planners Part 2 - Jason Haley, accessed on April 10, 2025,

https://jasonhaley.com/2024/05/27/semantic-kernel-hello-world-planners-part2/
43. Semantic Kernel - The new planners introduced in 1.0, accessed on April 10, 2025, https://www.developerscantina.com/p/semantic-kernel-new-planners/
44. Semantic Kernel - Stepwise Planner - Microsoft Developer Blogs, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/semantic-kernel-planners-stepwise-planner/
45. Semantic Kernel - Planner - The Developer's Cantina, accessed on April 10, 2025, https://www.developerscantina.com/p/semantic-kernel-planner/
46. Semantic Kernel Stepwise Planner | Restackio, accessed on April 10, 2025, https://www.restack.io/p/semantic-kernel-answer-stepwise-planner-cat-ai
47. The future of Planners in Semantic Kernel - Microsoft Developer Blogs, accessed on April 10, 2025, https://devblogs.microsoft.com/semantic-kernel/the-future-of-planners-in-semantic-kernel/
48. 15 - Stepwise Planner in Semantic Kernel #stepwise #planner #semantic - YouTube, accessed on April 10, 2025, https://www.youtube.com/watch?v=a1KhGKsigNo
49. Step by Step guide to develop AI Multi-Agent system using Microsoft Semantic Kernel and GPT-4o | by Akshay Kokane | Medium, accessed on April 10, 2025, https://medium.com/@akshaykokane09/step-by-step-guide-to-develop-ai-multi-agent-system-using-microsoft-semantic-kernel-and-gpt-4o-f5991af40ea6
50. Using Semantic Kernel to create multi-agent scenarios - The Developer's Cantina, accessed on April 10, 2025, https://www.developerscantina.com/p/semantic-kernel-multiagents/
51. Defining agents in a multi agent systems... | by RAJIB DEB | Medium, accessed on April 10, 2025, https://medium.com/@rajib76.gcp/defining-agents-in-a-multi-agent-systems-8dc2bc4744f7
52. Semantic Kernel Hello World Plugins Part 2 - Jason Haley, accessed on April 10, 2025, https://jasonhaley.com/2024/04/26/semantic-kernel-hello-world-plugin-part2/
53. Semantic Kernel - Native plugins - The Developer's Cantina, accessed on April 10, 2025, https://www.developerscantina.com/p/semantic-kernel-native-plugins/
54. Semantic Kernel Hello World Plugins Part 1 - Jason Haley, accessed on April 10, 2025, https://jasonhaley.com/2024/04/11/semantic-kernel-hello-world-plugin-part1/
55. semantic-kernel/python/samples/getting_started/06-memory-and-embeddings.ipynb at main, accessed on April 10, 2025, https://github.com/microsoft/semantic-kernel/blob/main/python/samples/getting_started/06-memory-and-embeddings.ipynb
56. Semantic Kernel: Using Memories to Create Intelligent AI Agents - Azure Feeds, accessed on April 10, 2025, https://azurefeeds.com/2024/10/07/semantic-kernel-using-memories-to-create-intelligent-ai-agents/

57. Semantic Kernel: Using Memories to Create Intelligent AI Agents - Jamie Maguire, accessed on April 10, 2025, https://jamiemaguire.net/index.php/2024/10/06/semantic-kernel-using-memories-to-create-intelligent-ai-agents/
58. semantic-kernel/docs/decisions/0050-updated-vector-store-design.md at main - GitHub, accessed on April 10, 2025, https://github.com/microsoft/semantic-kernel/blob/main/docs/decisions/0050-updated-vector-store-design.md
59. Azure Semantic Kernel Example with Embeddings and Pinecone - Stack Overflow, accessed on April 10, 2025, https://stackoverflow.com/questions/76417790/azure-semantic-kernel-example-with-embeddings-and-pinecone
60. Integrate Semantic Kernel with Astra DB Serverless - DataStax Docs, accessed on April 10, 2025, https://docs.datastax.com/en/astra-db-serverless/integrations/semantic-kernel.html
61. What are Semantic Kernel Vector Store connectors? (Preview) - Learn Microsoft, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/concepts/vector-store-connectors/
62. semantic-kernel-docs/semantic-kernel/concepts/vector-store-connectors/vector-search.md at main - GitHub, accessed on April 10, 2025, https://github.com/MicrosoftDocs/semantic-kernel-docs/blob/main/semantic-kernel/concepts/vector-store-connectors/vector-search.md
63. Overview | Kernel Memory - Microsoft Open Source, accessed on April 10, 2025, https://microsoft.github.io/kernel-memory/
64. microsoft/kernel-memory: RAG architecture: index and query any data using LLM and natural language, track sources, show citations, asynchronous memory patterns. - GitHub, accessed on April 10, 2025, https://github.com/microsoft/kernel-memory
65. Semantic Kernel Connectors Overview | Restackio, accessed on April 10, 2025, https://www.restack.io/p/semantic-kernel-answer-connectors-cat-ai
66. In-depth Semantic Kernel Demos | Microsoft Learn, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/get-started/detailed-samples
67. What are Semantic Kernel Vector Store connectors? (Preview ..., accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/concepts/vector-store-connectors/index
68. Legacy Semantic Kernel Memory Stores - Learn Microsoft, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/concepts/vector-store-connectors/memory-stores
69. Introduction to Memories in the Semantic Kernel SDK - YouTube, accessed on April 10, 2025, https://www.youtube.com/watch?v=nKjVrV23XN4
70. Semantic Kernel Text Memory Plugin | Restackio, accessed on April 10, 2025,

https://www.restack.io/p/semantic-kernel-answer-text-memory-plugin-cat-ai
71. Build a custom Copilot experience with your private data using and Kernel Memory, accessed on April 10, 2025, https://www.developerscantina.com/p/kernel-memory/
72. 5. Using Embeddings and Semantic Memory in Semantic Kernel Apps - YouTube, accessed on April 10, 2025, https://www.youtube.com/watch?v=FRf-7sv_ddU
73. kernel-memory in relationship with semantic-kernal #570 - GitHub, accessed on April 10, 2025, https://github.com/microsoft/kernel-memory/discussions/570
74. Leap Forward with Modern AI using Microsoft 365 Agents SDK and Semantic Kernel, accessed on April 10, 2025, https://arafattehsin.com/leap-forward-with-modern-ai-using-microsoft-365-agents-sdk-and-semantic-kernel/
75. Creating Plugins with the Semantic Kernel SDK and C# - DEV Community, accessed on April 10, 2025, https://dev.to/willvelida/creating-plugins-for-semantic-kernel-sdk-1ai4
76. Creating Plugins with the Semantic Kernel SDK and C# | Will Velida, accessed on April 10, 2025, https://www.willvelida.com/posts/create-plugins-semantic-kernel/
77. How to quickly start with Semantic Kernel | Microsoft Learn, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/get-started/quick-start-guide
78. semantic-kernel-docs/semantic-kernel/concepts/plugins/adding-native-plugins.md at main - GitHub, accessed on April 10, 2025, https://github.com/MicrosoftDocs/semantic-kernel-docs/blob/main/semantic-kernel/concepts/plugins/adding-native-plugins.md
79. Add native code as a plugin - Learn Microsoft, accessed on April 10, 2025, https://learn.microsoft.com/en-us/semantic-kernel/concepts/plugins/adding-native-plugins
80. AI Agent Deployment: Overcoming Technical Challenges - Rapid Innovation, accessed on April 10, 2025, https://www.rapidinnovation.io/post/for-developers-technical-challenges-and-solutions-in-ai-agent-deployment
81. Step-by-Step Guide to Building a Powerful AI Monitoring Dashboard with Semantic Kernel and Azure Monitor: Master TokenUsage Metrics and Custom Metrics using SK Filters | by Akshay Kokane | Medium, accessed on April 10, 2025, https://medium.com/@akshaykokane09/step-by-step-guide-to-building-a-powerful-ai-monitoring-dashboard-with-semantic-kernel-and-azure-7d4eed115d31
82. Building Multi-Agents with Semantic Kernel & Azure Durable Functions | #MVPConnect, accessed on April 10, 2025, https://www.youtube.com/watch?v=mOxPjNA3i7Y