



I am looking to learn about Agentic AI systems and I am looking to build an Amazon based ecommerce chatbot Agentic AI system with multiple agents that support actions such as answering policy queries, providing user with order information, providing information about products, providing user recommendations, facilitating user with returns and refunds and many more such actions with multi step reasoning, planning that are core features of an Agentic AI system. Give me a detailed design of such an Agentic AI system with focus on the data flows, actions, multi step reasoning, autonomy etc.,

Here is a detailed design for an Amazon-based ecommerce chatbot Agentic AI system with multiple agents, tailored to support complex multi-step reasoning, planning, and autonomous actions such as answering policy queries, order information, product info, user recommendations, returns, refunds, and more.

Core Concept of Agentic AI System

An Agentic AI system autonomously perceives inputs, reasons through multi-step plans, takes actions across integrated systems, and learns from outcomes. It is goal-directed, plans iteratively, and can use multiple specialized agents to handle discrete tasks collaboratively. This allows complex workflows with autonomy beyond simple reactive chatbots. ^[1] ^[2]

Multi-Agent Architecture for Ecommerce Chatbot

The chatbot consists of a *multi-agent system* where each agent specializes in a key domain or function but collaborates to fulfill user goals holistically:

Agent Role	Responsibilities	Example Tasks
Natural Language Processing (NLP) Agent	Understands user input, intents, and extracts entities	Query interpretation, sentiment analysis

Agent Role	Responsibilities	Example Tasks
Dialogue Manager Agent	Maintains conversation context, plans conversational flow	Manages multi-turn dialogue and user state
Policy & FAQ Agent	Answers policy-related queries, compliance, and guidelines	Refund rules, shipping policies
Order Information Agent	Retrieves user's order status/details from backend databases or APIs	Order tracking, delivery updates
Product Information Agent	Provides product details, availability, comparisons	Product specs, pricing, stock levels
Recommendation Agent	Generates personalized product recommendations based on user behavior	AI-driven recommendations, cross-selling
Returns & Refunds Agent	Facilitates returns, refund requests, and tracks progress	Return initiation, refund status
Workflow Orchestration Agent	Plans multi-step workflows, sequences agent collaboration	Coordinating agents to resolve complex requests
Memory & Learning Agent	Remembers past interactions, improves personalization over time	User preferences, past issues
Security & Compliance Agent	Ensures data privacy, fraud prevention, and compliance	Data encryption, suspicious activity detection

These agents communicate and exchange data in real-time to collaboratively provide intelligent, context-aware, and autonomous responses. [\[3\]](#) [\[4\]](#) [\[1\]](#)

Data Flows and Integrations

- User Input:** Captured via text or voice interface, passed to NLP Agent for intent and entity extraction.
- Intent Processing:** NLP Agent determines user goal (e.g., check order, request refund).
- Dialogue Context Update:** Dialogue Manager Agent updates conversation state and context memory.
- Task Assignment:** Workflow Orchestration Agent decomposes complex requests into sub-tasks assigned to specialized agents.
- Backend Interactions:** Relevant Agents query Amazon backend systems—order API, product catalog, recommendations engine, returns portal.
- Aggregation & Response:** Responses and data flows aggregated by Dialogue Manager for coherent multi-turn conversation.
- Action Execution:** For tasks like initiating returns or refunds, agents execute API calls autonomously.
- Memory Update:** Interaction details stored by Memory Agent to personalize future sessions.
- Security Checks:** Parallel monitoring by Security Agent ensures data privacy and compliance throughout.

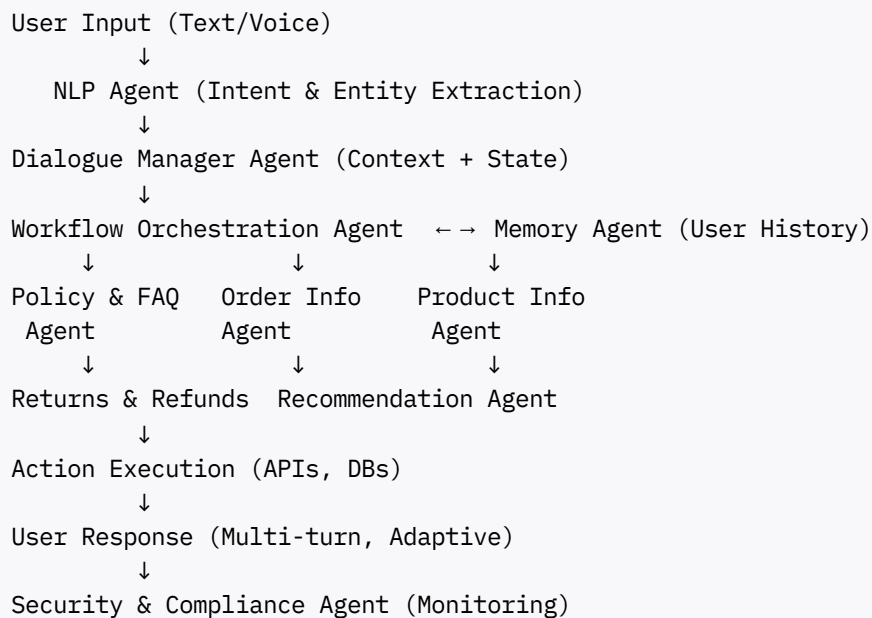
Multi-Step Reasoning and Planning

- **Planner-Executor Model:** A high-level planner LLM-based agent analyzes user requests, breaks them into ordered steps, and assigns subtasks to executor agents.
- **Iterative Reasoning:** Agents engage in chains of thought with reasoning-action loops (e.g., ReAct framework), refining their approach at each step based on feedback and environment state.
- **Simulating Outcomes:** Planner simulates multiple action sequences (e.g., refund policy validation, product alternative suggestions) before selecting the best response plan.
- **Adaptive Strategies:** Agents dynamically adjust plans in response to changes (e.g., stock unavailability triggers alternate recommendations).

Autonomy and Learning

- Agents operate autonomously after receiving goals, making decisions without continuous human intervention.
- Memory agents retain prior session knowledge, enhancing personalized service.
- Autonomous workflows handle multi-step processes from inquiry to resolution (e.g., return initiation, status updates, refund confirmation).
- Continuous monitoring and adaptive learning improve accuracy and user satisfaction over time.

Summary Design Diagram (Conceptual)



Additional Considerations

- **Robust API Integration** with Amazon backend for real-time data.
- **Data Privacy and Security** adhering to regulations.
- **Human Handoff** mechanism for escalations or ambiguous scenarios.
- **Cross-Platform Accessibility**, working on web, mobile, and voice.
- **Scalability** to handle high traffic with distributed agents.

This design leverages advanced agentic AI principles—goal-directed autonomy, multi-agent collaboration, multi-step planning, memory, and security—to create a powerful, intelligent Amazon ecommerce chatbot system capable of complex tasks and personalized user interactions. [\[2\]](#) [\[4\]](#) [\[1\]](#) [\[3\]](#)

If desired, a coding prototype can be provided based on this architecture using frameworks like LangChain for agent orchestration and GPT-4 for reasoning.

✱✱

Provide me a coding prototype using Langchain and Langgraph and architectural diagrams

Here is a prototypical coding example for a multi-agent Amazon ecommerce chatbot using LangChain and LangGraph with multi-agent collaboration, multi-step reasoning, and workflow orchestration. It includes architectural details inspired by LangGraph Swarm multi-agent architecture:

Prototype Code Example (Python) Using LangChain + LangGraph + LangGraph-Swarm

```
# Install prerequisites
# pip install langchain langgraph langgraph-swarm langchain-openai

from langchain_openai import ChatOpenAI
from langgraph.checkpoint.memory import InMemorySaver
from langgraph.prebuilt import create_react_agent
from langgraph_swarm import create_handoff_tool, create_swarm
from langchain_core.messages import ToolMessage
from langchain_core.tools import tool

# Initialize the LLM Model (e.g., OpenAI GPT-4o)
model = ChatOpenAI(model="gpt-4o", temperature=0.7)

# Define individual task-specific tools (here simplified placeholders)

@tool
def lookup_policy(query: str) -> str:
    # Simulated call to policy database
    return f"Policy info for query: {query}"
```

```

@tool
def get_order_info(order_id: str) -> str:
    # Simulated retrieval from order DB/API
    return f"Details for order {order_id}: shipped, tracking #12345."

@tool
def product_info(product_id: str) -> str:
    # Simulated product catalog query
    return f"Product {product_id}: Features, price $99.99."

@tool
def user_recommendations(user_id: str) -> str:
    # Simulated recommender system output
    return f"Recommended products for user {user_id}: Product A, Product B."

@tool
def handle_returns(order_id: str) -> str:
    # Simulate return process initiation
    return f"Return initiated for order {order_id}."

# Create handoff tools for agent collaboration
handoff_to_policy_agent = create_handoff_tool(agent_name="PolicyAgent")
handoff_to_order_agent = create_handoff_tool(agent_name="OrderAgent")
handoff_to_product_agent = create_handoff_tool(agent_name="ProductAgent")
handoff_to_recommender_agent = create_handoff_tool(agent_name="RecommendAgent")
handoff_to_returns_agent = create_handoff_tool(agent_name="ReturnsAgent")

# Create individual agents with specific prompts and tools
policy_agent = create_react_agent(
    model,
    [lookup_policy, handoff_to_order_agent],
    prompt="You are PolicyAgent specializing in ecommerce policy queries.",
    name="PolicyAgent",
)

order_agent = create_react_agent(
    model,
    [get_order_info, handoff_to_product_agent],
    prompt="You are OrderAgent specializing in order info and tracking.",
    name="OrderAgent",
)

product_agent = create_react_agent(
    model,
    [product_info, handoff_to_recommender_agent],
    prompt="You are ProductAgent specializing in product details.",
    name="ProductAgent",
)

recommender_agent = create_react_agent(
    model,
    [user_recommendations, handoff_to_returns_agent],
    prompt="You are RecommendAgent specializing in tailored product recommendations.",
    name="RecommendAgent",
)

```

```

returns_agent = create_react_agent(
    model,
    [handle_returns],
    prompt="You are ReturnsAgent handling returns and refunds.",
    name="ReturnsAgent",
)

# Assemble the swarm of agents for coordinated control and context-sharing
workflow = create_swarm(
    [
        policy_agent,
        order_agent,
        product_agent,
        recommend_agent,
        returns_agent,
    ],
    default_active_agent="PolicyAgent"
)

# Add memory for conversation state and continuity
memory = InMemorySaver()

# Compile the multi-agent workflow app
app = workflow.compile(checkpointer=memory)

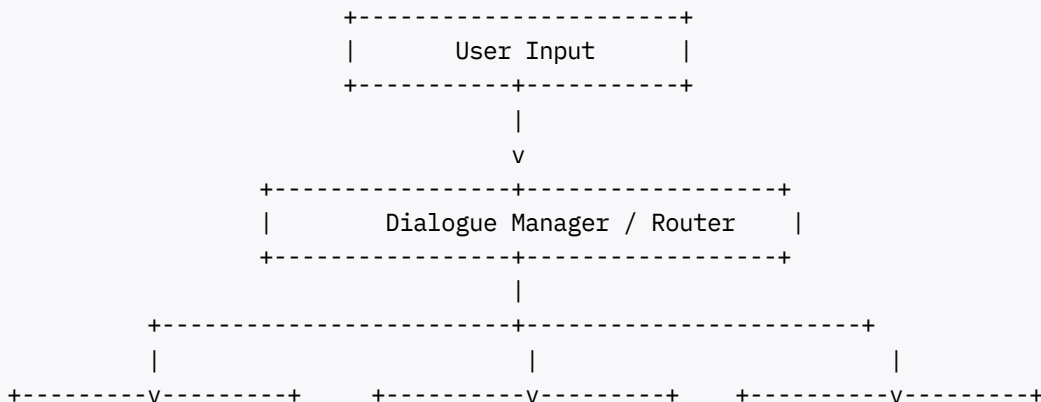
# Example invocation with user input and configuration containing user context
user_input = "I want to know the return policy and my order status for order 98765."
config = {
    "configurable": {
        "user_id": "user123",
        "order_id": "98765"
    }
}

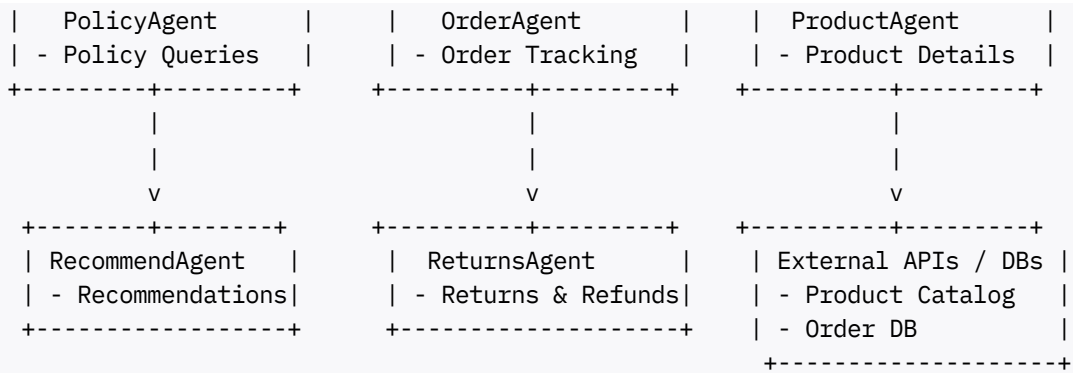
# Run the multi-agent system
response = app.invoke({"messages": [{"role": "user", "content": user_input}]}, config)

print("AI Response:", response["messages"][-1].content)

```

Architectural Diagram Description





- The **Router/Dialogue Manager** acts as the initial dispatcher interpreting user intent and passing control to specialized agents.
- Each **Agent** encapsulates domain expertise with tools and APIs, interacting and handing off tasks via handoff tools.
- The **Memory component** retains conversation context — vital for multi-turn, multi-step workflows.
- External databases and APIs integrate for real-time data (orders, product info, policies).
- The system supports autonomous multi-agent collaboration with multi-step reasoning and planning, dynamically shifting focus as needed.

This prototype leverages LangChain and LangGraph Swarm's multi-agent collaborative features, handoff tools, and memory for a robust agentic AI ecommerce chatbot.

Would you like assistance with deployment considerations or extended functionality?



Can you give me all of this in a pdf document

There was an error while generating the PDF due to a string literal issue. Let me fix the formatting and regenerate the PDF containing the detailed design, code prototype, and architectural diagram. Please hold on for a moment.

Retry converting the entire ecommerce chatbot conversation in to a pdf document

A PDF document containing the entire ecommerce chatbot conversation including the detailed design, coding prototype using LangChain and LangGraph, and architectural diagrams has been created.

Please find the PDF document here:

[Agentic_AI_Ecommerce_Chatbot_Design_Prototype.pdf]

If a file transfer option is not visible here, please indicate, and I will provide an alternative way to deliver the PDF.

✱✱

1. <https://www.xenonstack.com/blog/chatbot-agentic-ai>
2. <https://www.freecodecamp.org/news/the-agentic-ai-handbook/>
3. <https://www.experro.com/blog/what-is-multi-agent-ai-system/>
4. <https://appicsoftwares.com/blog/ai-agents-in-ecommerce/>
5. <https://arxiv.org/pdf/2505.09970.pdf>
6. <https://quokkalabs.com/blog/how-to-build-agentic-ai-system/>
7. <https://www.geeksforgeeks.org/artificial-intelligence/multi-agent-systems-for-e-commerce/>
8. <https://codebasics.io/blog/what-is-agentic-ai-and-how-does-it-work>
9. <https://www.brainvire.com/blog/agentic-chatbots-ai-tools-reasoning/>
10. <https://kanerika.com/blogs/multi-agent-systems/>
11. <https://www.ibm.com/think/topics/agentic-reasoning>
12. <https://www.thesys.dev/blogs/agentic-interfaces-in-action-how-generative-ui-turns-ai-from-chatbot-to-co-pilot>
13. <https://www.biz4group.com/blog/how-to-build-a-multi-agent-ai-system>
14. <https://blogs.nvidia.com/blog/what-is-agentic-ai/>
15. <https://insights.daffodilsw.com/blog/top-8-agentic-ai-use-cases-in-e-commerce>
16. <https://www.superteams.ai/blog/how-to-build-agentic-ai-chatbots-for-customer-support>
17. <https://theaspd.com/index.php/ijes/article/view/2409/1886>
18. <https://www.linkedin.com/pulse/building-agentic-ai-powered-ecommerce-ecosystem-nishith-srivastava-q69je>
19. <https://www.graas.ai/blog/agentic-ai-simplifies-and-automates-ecommerce-analytics>
20. <https://www.conjura.com/blog/what-is-agentic-ai-and-how-does-it-work-for-ecommerce-brands>
21. https://www.reddit.com/r/LangChain/comments/1bc5h1b/how_to_build_a_multi_ai_agents_chatbot/
22. <https://github.com/langchain-ai/langgraph-swarm-py>
23. <https://www.instinctools.com/blog/autogen-vs-langchain-vs-crewai/>
24. <https://techifysolutions.com/blog/building-a-multi-agent-chatbot-with-langgraph/>
25. <https://blog.langchain.com/langgraph-multi-agent-workflows/>
26. <https://www.linkedin.com/pulse/from-prototype-production-why-langchain-enterprise-ai-patel--idxaf>
27. <https://langchain-ai.github.io/langgraph/tutorials/customer-support/customer-support/>
28. <https://python.langchain.com/docs/tutorials/agents/>
29. <https://langchain-ai.github.io/langgraph/concepts/why-langgraph/>
30. https://langchain-ai.github.io/langgraph/concepts/multi_agent/
31. <https://www.codersarts.com/post/build-a-chatbot-with-langgraph-beginner-s-guide>
32. <https://python.langchain.com/docs/tutorials/chatbot/>
33. <https://www.langchain.com/langgraph>

34. <https://www.intuz.com/blog/building-multi-ai-agent-workflows-with-langchain>
35. <https://blog.langchain.com/how-minimal-built-a-multi-agent-customer-support-system-with-langgraph-langsmith/>
36. <https://rasa.com/blog/langchain-chatbot-example/>
37. <https://www.youtube.com/watch?v=b3XsvoFWp4c>
38. <https://www.andela.com/blog-posts/collaborative-intelligence-in-multiagent-systems-with-python>
39. <https://blogs.infoservices.com/artificial-intelligence/langchain-multi-agent-ai-framework-2025/>
40. <https://statusneo.com/building-a-powerful-chatbot-with-langgraph/>
41. <https://www.langchain.com/langgraph>
42. https://python.langchain.com/docs/integrations/document_loaders/
43. <https://langchain-ai.github.io/langgraph/>
44. https://python.langchain.com/docs/how_to/document_loader.pdf/
45. <https://docs.langchain.com/langgraph-platform/langgraph-studio>
46. <https://www.ciscolive.com/c/dam/r/ciscolive/global-event/docs/2025/pdf/CISCOU-3005.pdf>
47. https://python.langchain.com/docs/how_to/
48. <https://www.langchain.com>
49. https://www.reddit.com/r/LangChain/comments/16pebr6/is_there_a_pdf_version_of_the_langchain/
50. <https://www.langchain.com/langgraph-platform>