Cognitive AI for the Future: Multimodal Models and RAG in Vision Language Applications, from Training to Deployment

OpenVINO™

CVPR 2025

Module 4 :

Build Your Own AI Assistant with Multi-agent Workflow and Multimodal RAG

Speaker: Tiep Le

Job title: AI Research Scientist

Outline (30min)

- **Overview Multi-agent Workflow with Multimodal RAG** 10m
- Multi-agent Workflow with MCP 15m
- Demo 5m

LLM Agent

- An agent is a system that uses an LLM to decide the **control flow of an application**.
  - An LLM can route between few potential paths.
  - An LLM can decide which of many tools to call.
  - An LLM can decide whether the generated answer is sufficient or more work is needed.

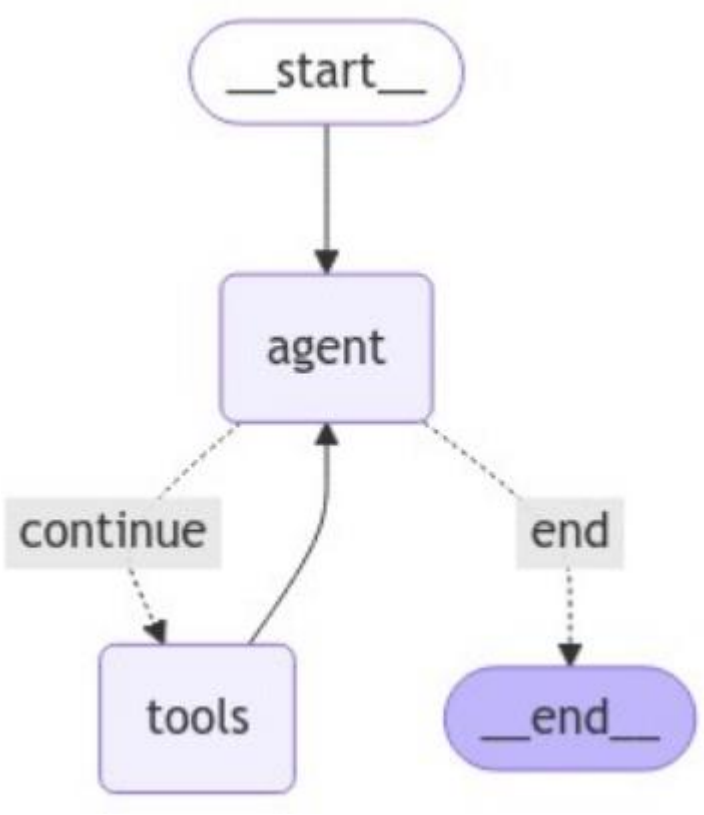
https://langchain-ai.github.io/langgraph/concepts/agentic_concepts/

LLM Agent

- An LLM agent should include 3 core functionalities:
  - Tool calling: Allowing LLM to select and use various tools as needed.
  - Memory: Enabling the agent to retain and use information from previous steps or conversation history.
  - Planning: Empowering the LLM to create and follow multi-step plans to achieve goals.

https://langchain-ai.github.io/langgraph/concepts/agentic_concepts/

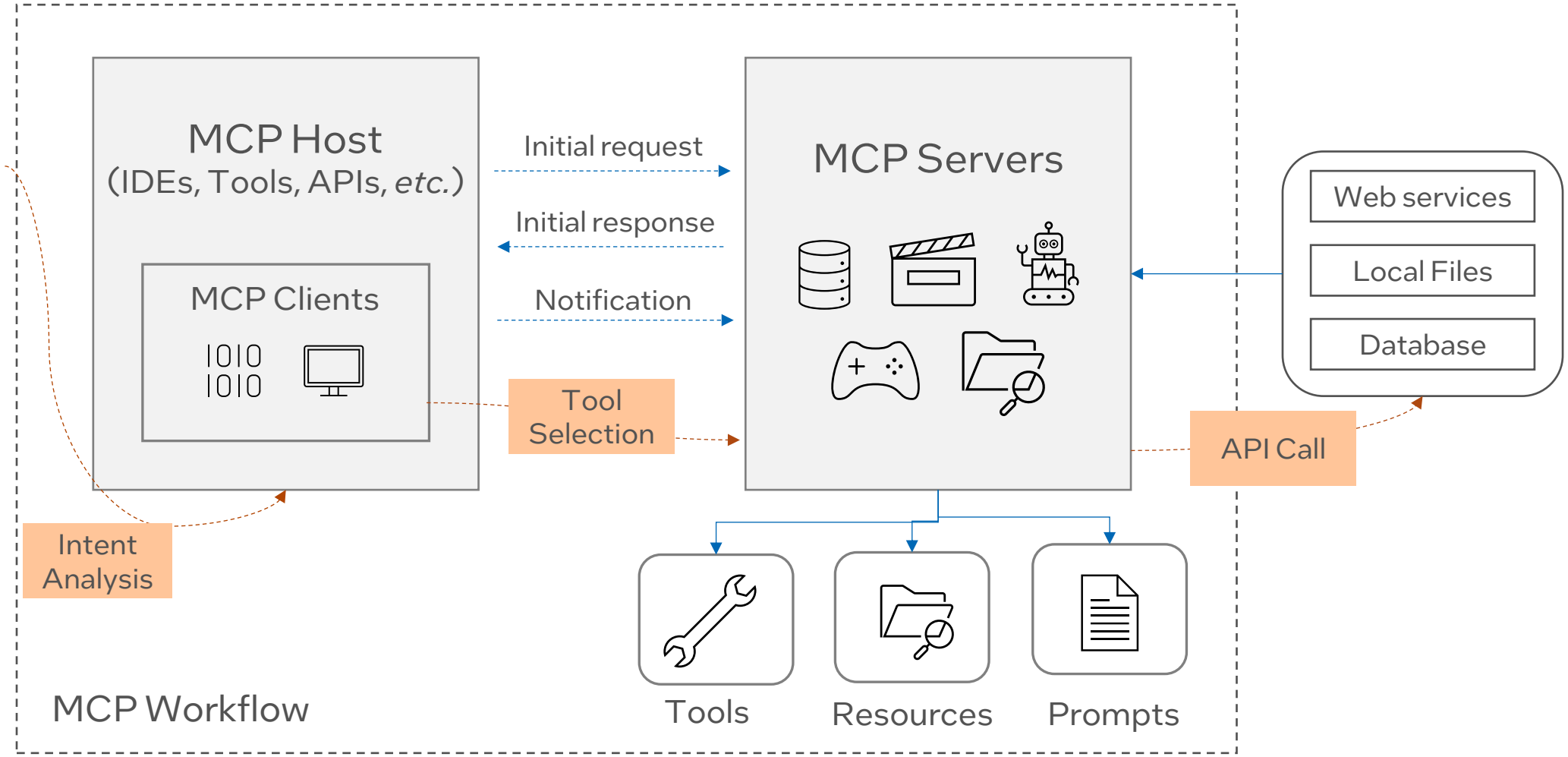
ReAct Agent (Reason + Act)




```
You are designed to help with a variety of tasks.

## Tools

You have access to the following tools:
{tool_desc}

## Output Format
To answer the question, please use the following format.

```
Thought: I need to use a tool to help me answer the question.
Action: tool name (one of {tool_names}) if using a tool.
Action Input: the input to the tool, in a JSON format representing the kwargs (
e.g. {{"input": "hello world", "num_beams": 5}})
```

If this format is used, the user will respond in the following format:

```
Observation: tool response
```

You should keep repeating the above format until you have enough information to
answer the question without using any more tools. At that point, you MUST respond
in the one of the following two formats:

```
Thought: I can answer without using any more tools.
Answer: [your answer here]
```

```
Thought: I cannot answer the question with the provided tools.
Answer: Sorry, I cannot answer your query.
```
```

https://docs.llamaindex.ai/en/stable/examples/agent/react_agent/
https://langchain-ai.github.io/langgraph/how-tos/react-agent-from-scratch/
https://arxiv.org/abs/2210.03629

Model Context Protocol (MCP)



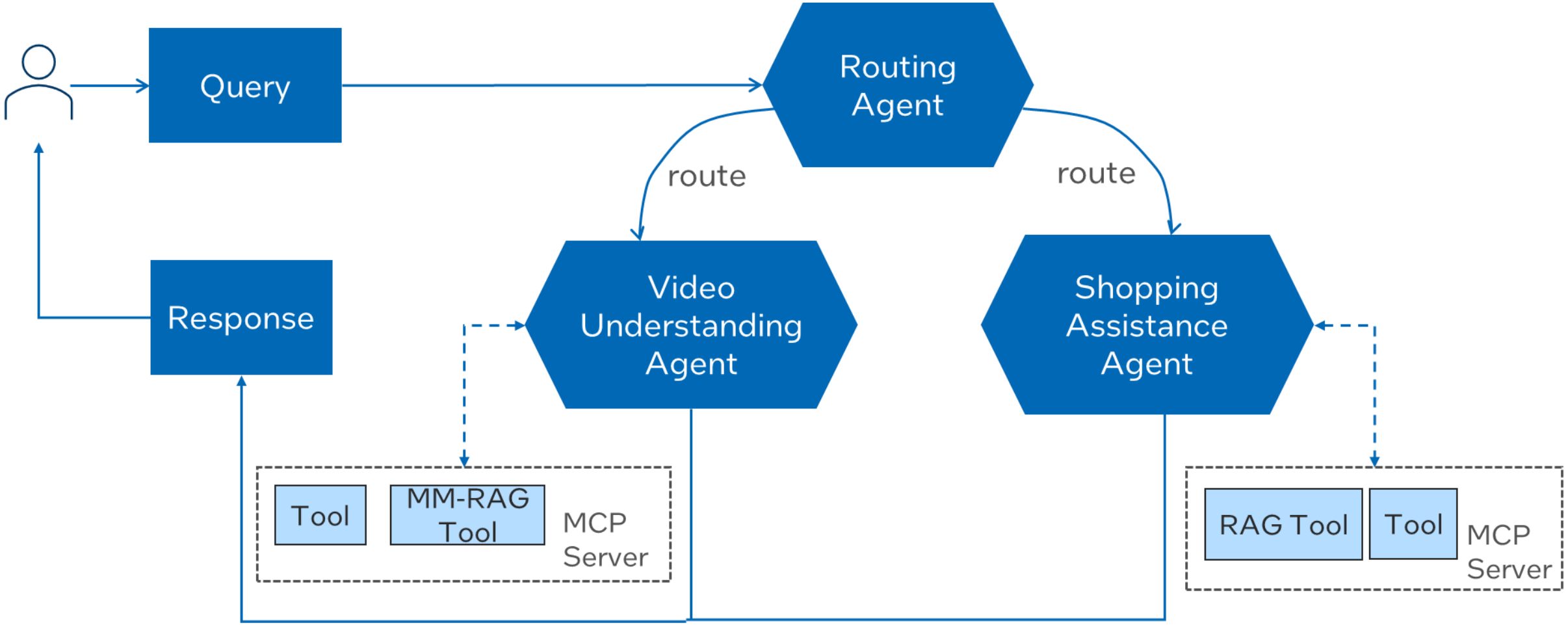
Multi-agent Workflow

- Multi-agent are multiple independent LLM agents connected in a specific way.
  - Each LLM agent can have its own prompt and tools.
  - Each LLM agent can collaborate with other agents in a specific way.
- Two main considerations when designing multi-agent workflows
  - What are the multiple independent LLM agents?
  - How are those agents connected?

https://blog.langchain.dev/langgraph-multi-agent-workflows/

Multi-agent Workflow with Multimodal RAG & OpenVINO

Run on local machine



Outline (30min)

- Overview Multi-agent Workflow with Multimodal RAG 8m
- **Multi-agent Workflow with MCP** 15m
- Demo 5m

Example MCP Server for Smart Retail

```python
from mcp.server.fastmcp import FastMCP
from pydantic import BaseModel
import math

mcp = FastMCP("Smart Retail Tools")

# In-memory cart storage
_cart_items = []

# --- Tool: Clear Cart ---
@mcp.tool()
def clear_cart() -> str:
    """
    Use this tool to clear all items from the shopping cart.

    Returns:
    - Confirmation message.
    """
    _cart_items.clear()
    return "Shopping cart has been cleared"

# Expose the FastAPI app
mcp.run(transport="sse")
```

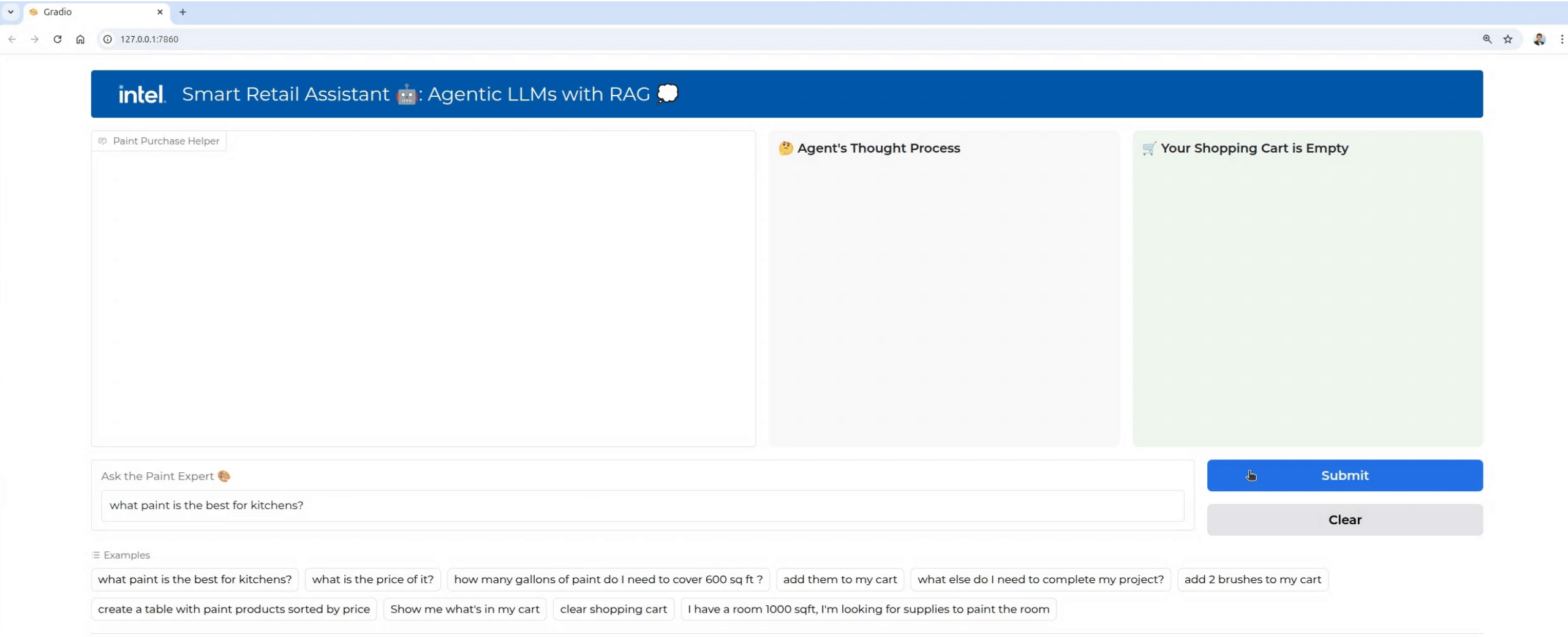
MCP Clients with LlamaIndex Connector

```python
from llama_index.tools.mcp import BasicMCPClient, McpToolSpec

# assuming the video search MCP server is run at http://localhost:3000/
mcp_client = BasicMCPClient("http://127.0.0.1:3000/sse")
mcp_tool = McpToolSpec(client=mcp_client)
video_search_tools = await mcp_tool.to_tool_list_async()
```

Multi-agent Workflow with LlamaIndex

```python
from llama_index.core.agent.workflow import ReActAgent
from llama_index.core.agent.workflow import AgentWorkflow

video_search_agent = ReActAgent(
    name="VideoSearchAgent",
    description="Useful for answering question that requires to search from video.",
    llm=llm,
    tools=video_search_tools,
)
shopping_cart_agent = ReActAgent(
    name="ShoppingCartAgent",
    description="Useful for reponding to requests that inquiry about shopping cart, product specification.",
    llm=llm,
    tools=shopping_cart_tools,
)
routing_agent = ReActAgent(
    name="RoutingAgent",
    description="Useful for routing the query appropriately to either video search agent or shopping cart agent",
    system_prompt=(
        "You are the Routing Agent that will analyze the request carefully "
        "and determine which agent you should hand off the control to.\n"
        "- You should hand off the control to the Video Search Agent "
        "if the query requires search and understanding from the video.\n"
        "- You should hand off the control to the Shopping Cart Agent "
        "if the query inquiries product specification "
        "and detailed information of shopping cart, and requires update the cart.\n"
        "You must hand off the control to either the Video Search Agent or the Shopping Cart Agent."
    ),
    llm=llm,
    tools=None,
    can_handoff_to=["VideoSearchAgent, ShoppingCartAgent"],
)

multiagent_workflow = AgentWorkflow(
    agents=[video_search_agent, shopping_cart_agent, routing_agent],
    root_agent=routing_agent.name,
)

handler = await agent_workflow.run(
    user_msg="Show me my cart."
)
```

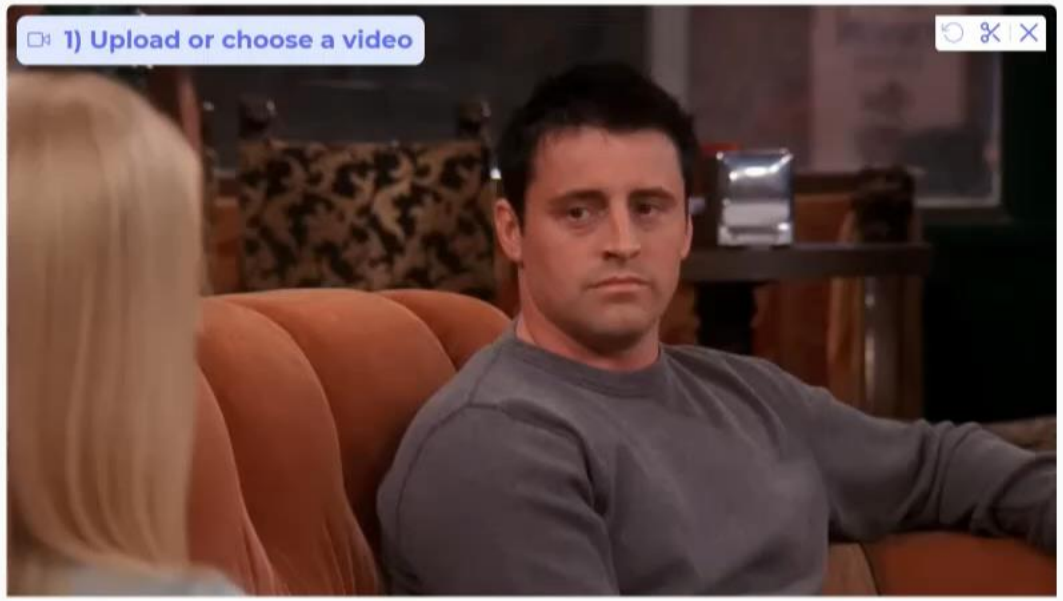
Agentic Workflow with RAG & MCP



127.0.0.1:7875

Powered by OpenVINO + MCP Tools

1) Upload or choose a video



2) Build Vector Store

Vector Store is Ready

Agent's Reasoning Log

Your Actions / Cart

Conversation

Type your message...

Send

Stop

Clear

Click example, then Send

What dessert is included in this video?

Conclusion

Benefits of Agentic Multimodal RAG with OpenVINO

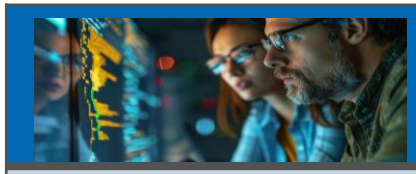
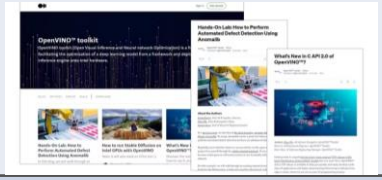
- **High-Performance Inference with OpenVINO:** Simple steps to optimize and deploy on local machines

- **Modular Agentic Workflow with MCP**: Easy to connect models, tools, and data sources in a flexible pipeline

- **Multimodal Reasoning Made Practical**: Multimodal data to enable rich video Q&A and intelligent task execution

- **From Perception to Action**: OpenVINO empowers automating tasks like product search and online purchasing with smooth, low-latency interactions

OpenVINO™ Developer Resources

	OpenVINO tutorial sample codes: https://github.com/openvinotoolkit/openvino_notebooks
	OpenVINO DevCon Workshops: https://bizwebcast.intel.cn/devcon2025.aspx
	Technical blogs: https://medium.com/@openvino
	AI reference kits: https://github.com/openvinotoolkit/openvino_build_deploy/tree/master/ai_ref_kits
	OpenVINO GenAI API source codes & samples https://github.com/openvinotoolkit/openvino.genai
	OpenVINO website with full information： openvino.ai

Try It Yourself

openvino.ai

OpenVINO™

intel

Connect With Us

OpenVINO

intel

Notices and Disclaimers

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.

Performance results are based on testing as of dates shown in configurations and may not reflect all publicly available updates. See backup for configuration details.

Intel technologies may require enabled hardware, software or service activation.

Your costs and results may vary.

Intel is committed to respecting human rights and avoiding complicity in human rights abuses. See Intel's Global Human Rights Principles. Intel's products and software are intended only to be used in applications that do not cause or contribute to a violation of an internationally recognized human right.

Thank You

intel ai