Mar 16, 2025

Advanced Techniques to Build Robust Agentic Systems (Part B)

AI Agents Crash Course—Part 6 (with implementation).

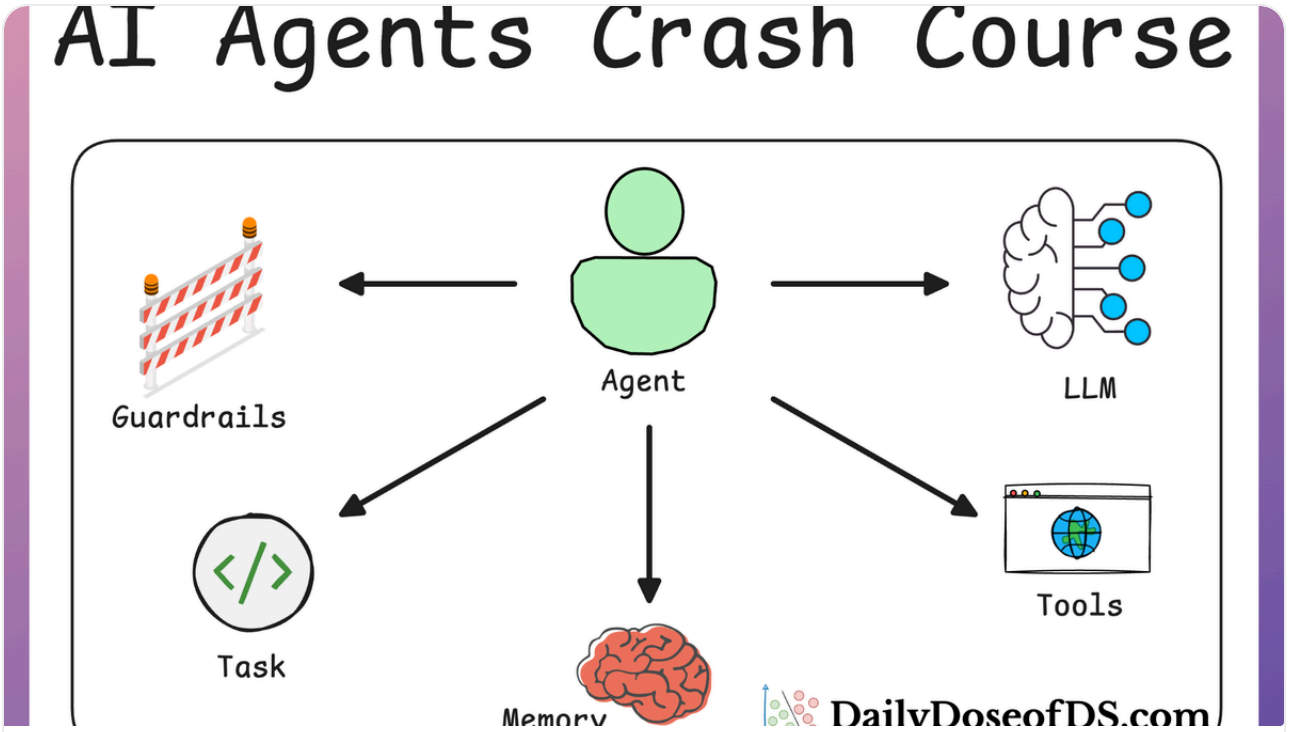
Avi Chawla, Akshay Pachaar

Introduction

In Part 5, we moved into build learning advanced techniques that make AI agents more robust, dynamic, and adaptable.

- Guardrails → Enforcing constraints to ensure agents produce reliable and expected outputs.
- Referencing other Tasks and their outputs → Allowing agents to dynamically use previous task results.
- Executing tasks async → Running agent tasks concurrently to optimize performance.
- Adding callbacks → Allowing post-processing or monitoring of task completions.
- Introduce human-in-the-loop during execution → Introducing human-in-the-loop mechanisms for validation and control.
- Hierarchical Agentic processes → Structuring agents into sub-agents and multi-level execution trees for more complex workflows.
- Multimodal Agents → Extending CrewAI agents to handle text, images, audio, and beyond.
- And more.

We split these into two parts to improve the reading experience and ensure that you do not feel overwhelmed with several details.



AI Agents Crash Course—Part 5 (With Implementation)

A deep dive into advanced techniques to make robust Agentic systems.

Daily Dose of Data Science • Avi Chawla

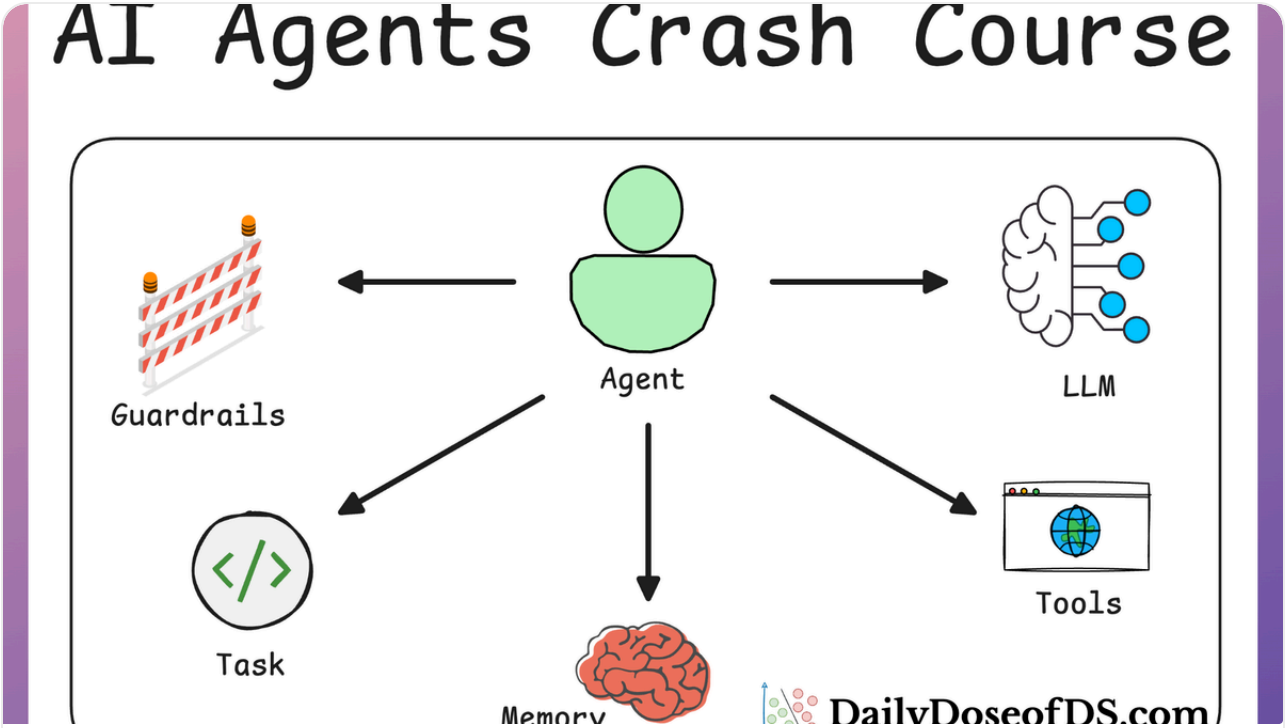
We have already covered half of this in Part 5 and in Part 6 (this article), we shall be covering the rest.

Let's dive in!

On a side note:

If you haven't read Part 1 to Part 5 yet, we highly recommend doing so before moving ahead.

- In Part 1, we covered the fundamentals of Agentic systems, understanding how AI agents can act autonomously to perform structured tasks.

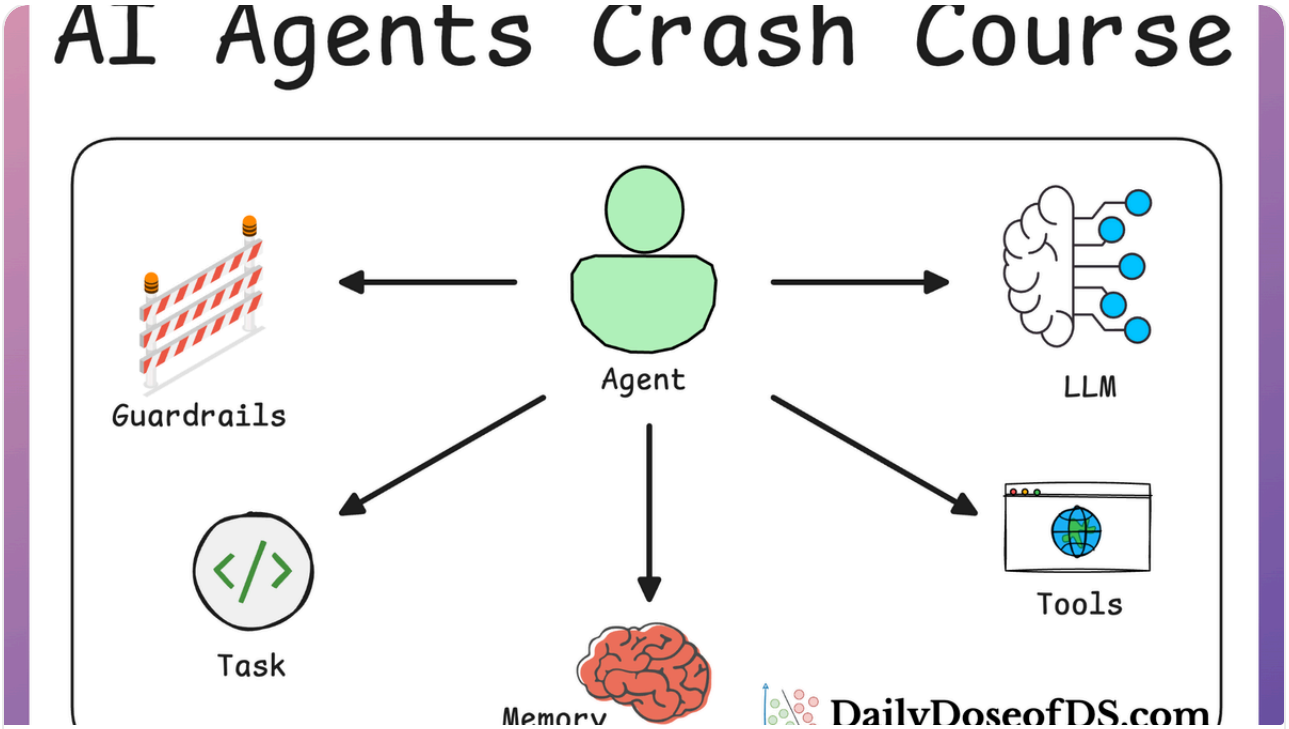


AI Agents Crash Course—Part 1 (With Implementation)

A deep dive into the fundamentals of Agentic systems, building blocks, and how to build them.

Daily Dose of Data Science • Avi Chawla

- In Part 2, we explored how to extend Agent capabilities by integrating custom tools, using structured tools, and building modular Crews to compartmentalize responsibilities.

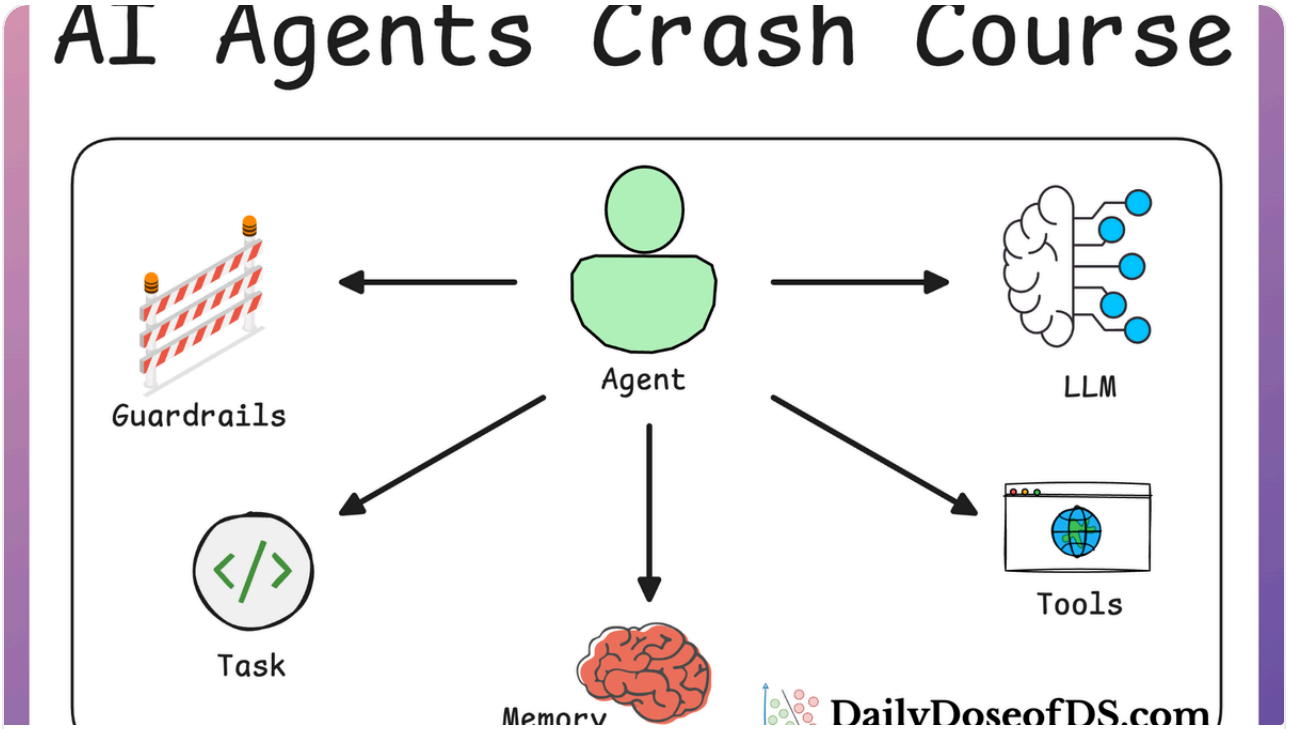


AI Agents Crash Course—Part 2 (With Implementation)

A deep dive into the fundamentals of Agentic systems, building blocks, and how to build them.

Daily Dose of Data Science • Avi Chawla

- In Part 3, we focused on Flows, learning about state management, flow control, and integrating a Crew into a Flow. As discussed last time, with Flows, you can create structured, event-driven workflows that seamlessly connect multiple tasks, manage state, and control the flow of execution in your AI applications.

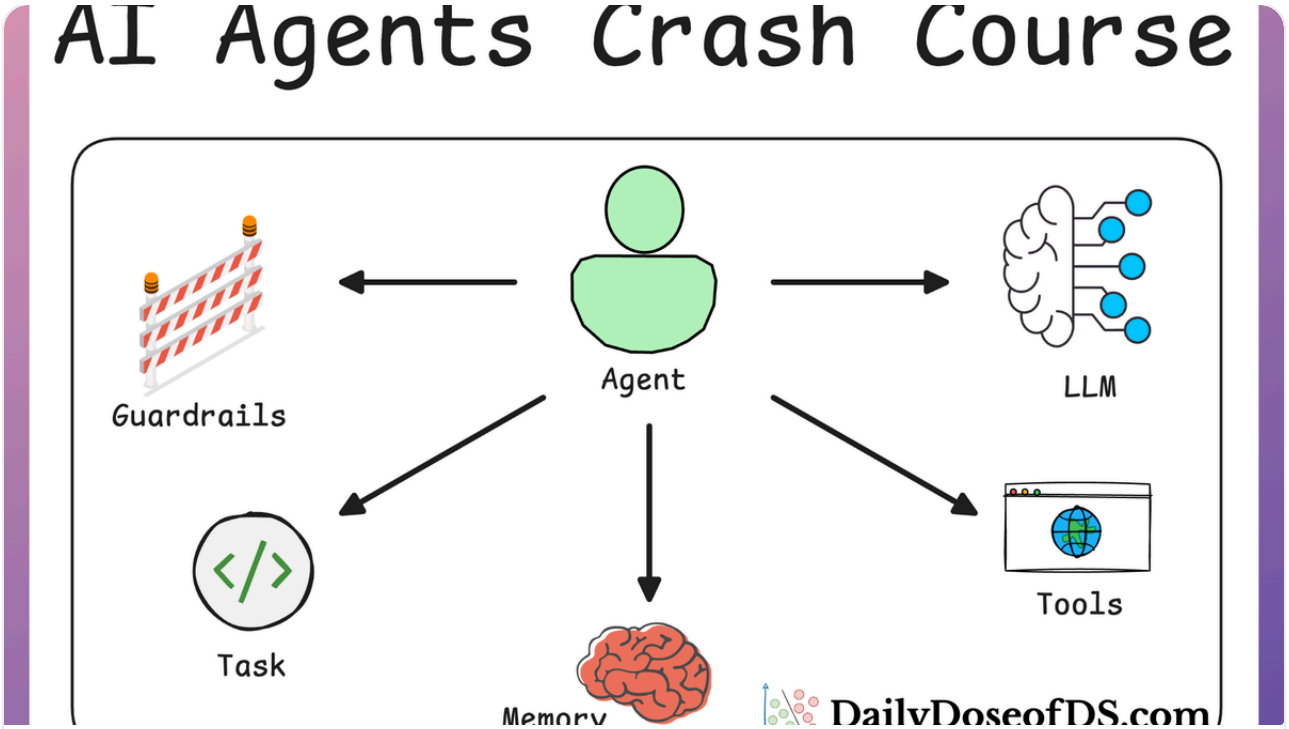


AI Agents Crash Course—Part 3 (With Implementation)

A deep dive into implementing Flows for building robust Agentic systems.

Daily Dose of Data Science • Avi Chawla

- In Part 4, we extended these concepts into real-world multi-agent, multi-crew Flow projects, demonstrating how to automate complex workflows such as content planning and book writing.



AI Agents Crash Course—Part 4 (With Implementation)

A deep dive into implementing Flows for building robust Agentic systems.

Daily Dose of Data Science • Avi Chawla

Now, in Part 5 and 6, we shall move into advanced techniques that make AI agents more robust, dynamic, and adaptable.

Installation and setup

💡 Feel free to skip this part if you have followed these instructions before.

Throughout this crash course, we have been using CrewAI, an open-source framework that makes it seamless to orchestrate role-playing, set goals, integrate tools, bring any of the popular LLMs, etc., to build autonomous AI agents.

GitHub - crewAIInc/crewAI: Framework for orchestrating role-playing, autonomous AI agents. By fostering collaborative intelligence, CrewAI empowers agents to work together seamlessly, tackling complex tasks.

Framework for orchestrating role-playing, autonomous AI agents. By fostering collaborative intelligence, CrewAI empowers agents to work together seamlessly, tackling...

GitHub • crewAIInc

To highlight more, CrewAI is a standalone independent framework without any dependencies on Langchain or other agent frameworks.

Let's dive in!

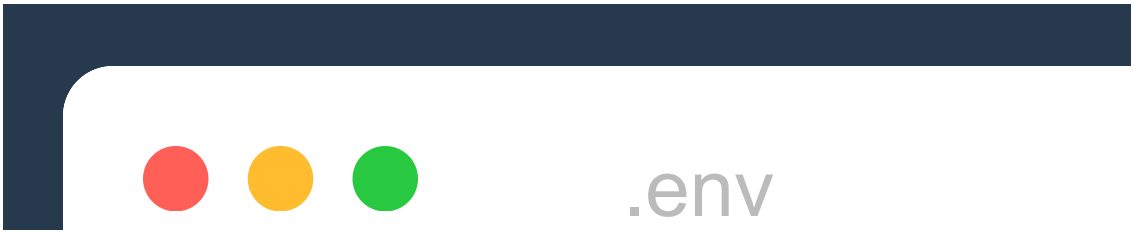
To get started, install CrewAI as follows:

Like the RAG crash course, we shall be using Ollama to serve LLMs locally. That said, CrewAI integrates with several LLM providers like:

- OpenAI
- Gemini
- Groq
- Azure
- Fireworks AI
- Cerebras
- SambaNova
- and many more.

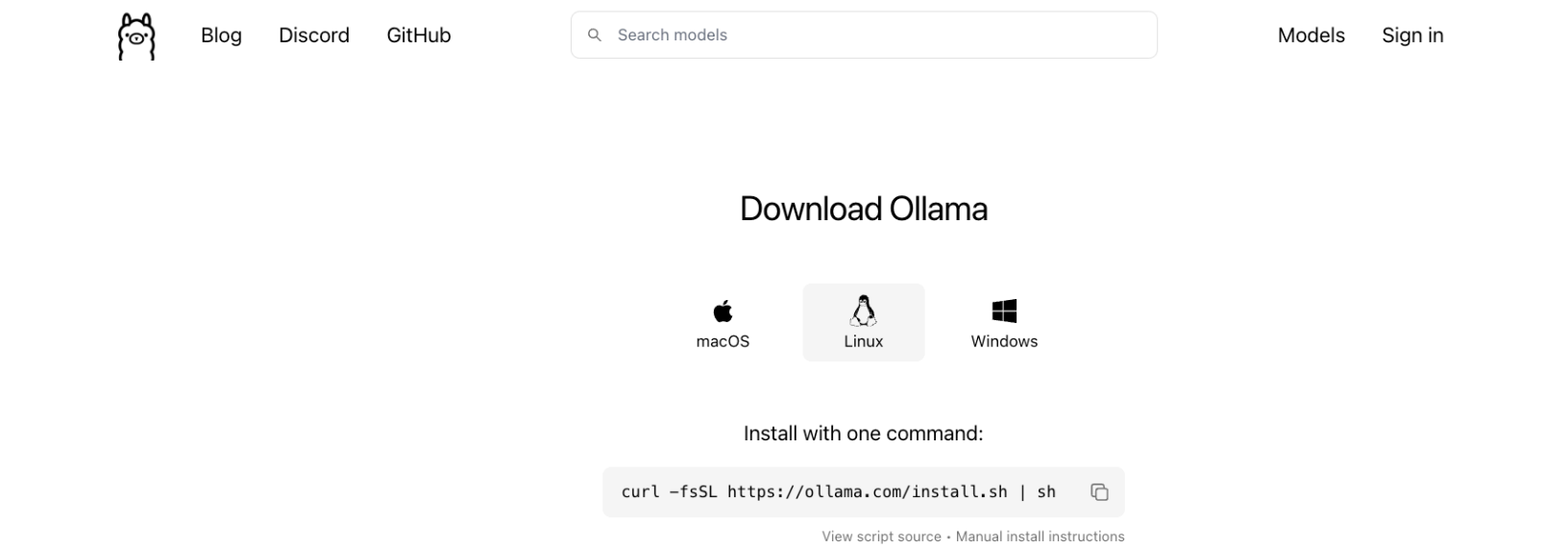
> 💡 If you have an OpenAI API key, we recommend using that since the outputs may not make sense at times with weak LLMs. If you don't have an API key, you can get some credits by creating a dummy account on OpenAI and use that instead. If not, you can continue reading and use Ollama instead but the outputs could be poor in that case.

To set up OpenAI, create a `.env` file in the current directory and specify your OpenAI API key as follows:

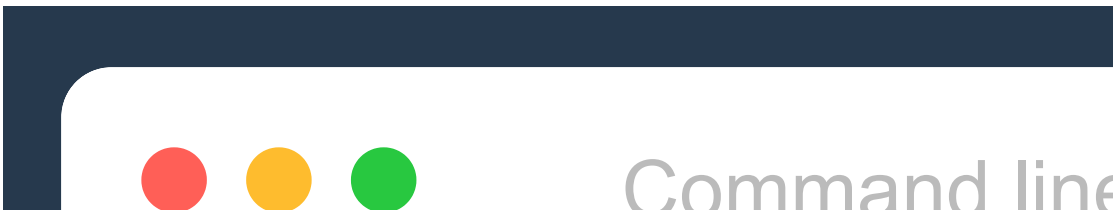


Also, here's a step-by-step guide on using Ollama:

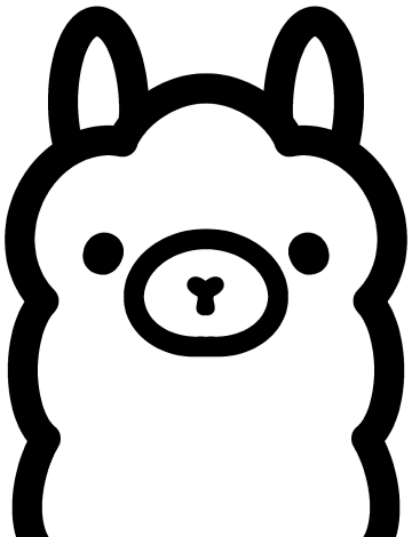
- Go to **Ollama.com**, select your operating system, and follow the instructions.



- If you are using Linux, you can run the following command:



- Ollama supports a bunch of models that are also listed in the model library:



library

Get up and running with large language models.

Blog Discord GitHub Search models Models Sign in Download

Models

Filter by name... Most popular

llama3.2

Meta's Llama 3.2 goes small with 1B and 3B models.

tools 1b 3b

2.2M Pulls 63 Tags Updated 5 weeks ago

llama3.1

Llama 3.1 is a new state-of-the-art model from Meta available in 8B, 70B and 405B parameter sizes.

tools 8b 70b 405b

8M Pulls 93 Tags Updated 7 weeks ago

gemma2

Google Gemma 2 is a high-performing and efficient model available in

Once you've found the model you're looking for, run this command in your terminal:

Command line

```
# Replace 'llama3.2' with the desired model name

ollama run llama3.2
```

The above command will download the model locally, so give it some time to complete. But once it's done, you'll have Llama 3.2 3B running locally, as shown below, which depicts Microsoft's Phi-3 served locally through Ollama:

That said, for our demo, we would be running Llama 3.2 1B model instead since it's smaller and will not take much memory:

Done!

Everything is set up now and we can move on to building robust agentic workflows.

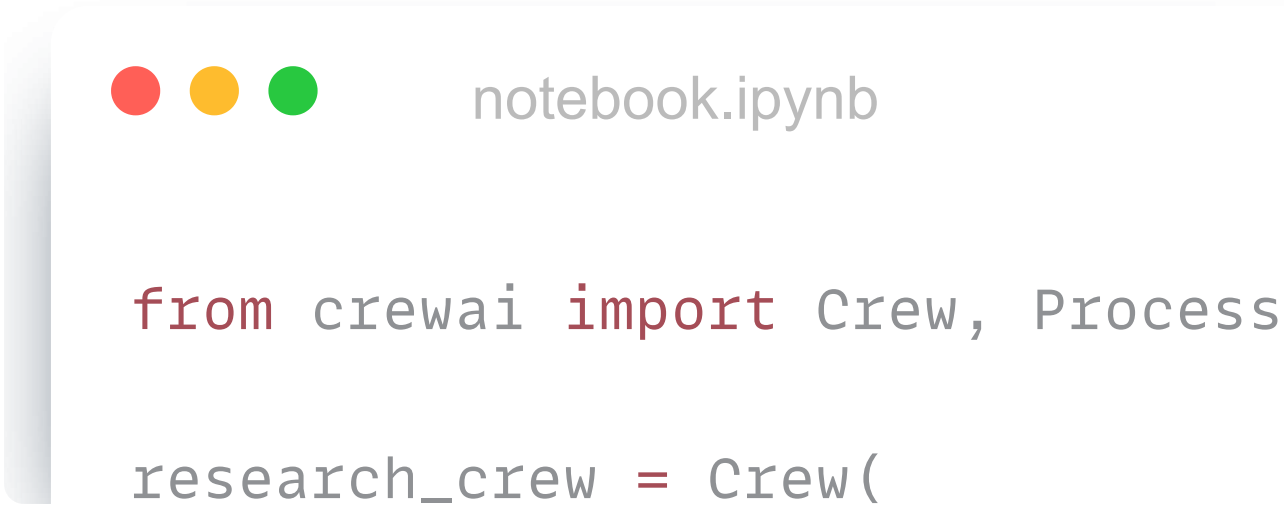
You can download the code for this article below:

notebook

notebook.ipynb • 28 KB

Hierarchical processes

The Crews we have primarily built and discussed so far were Sequential Crews—one task is executed after another, and so on, and we



However, in several real-world organizations, tasks aren't just executed sequentially—there's a hierarchical structure where managers delegate work, ensure quality control, and oversee task execution.

CrewAI allows us to replicate this management structure by using a hierarchical process, where a manager agent oversees specialized agents, assigns tasks, and validates outcomes.

The hierarchical process enables structured AI workflows, where a manager agent ensures task execution meets required standards.

Here's what happens inside:

- Task delegation → The manager assigns tasks based on agent expertise.
- Result validation → Ensures AI-generated outputs meet quality requirements.

In this following walkthrough, we'll build a hierarchical AI research workflow that:

- Breaks project research into distinct tasks: Market Demand, Risk Analysis, and Return on Investment.
- Delegates tasks to specialized AI agents, ensuring accurate and structured results.
- Uses a Manager AI to review and compile the final report, ensuring quality control.

Here's a step-by-step workflow:

- The Manager AI assigns the project research tasks.
- Three specialized agents conduct Market Demand, Risk Analysis, and ROI Research.
- The Manager AI reviews their outputs and compiles a final research report.

One thing to note before we dive into the implementation is that in a hierarchical workflow, we have two types of agents:

- The Manager Agent → Oversees the entire workflow, delegates tasks, and reviews outputs.
- Specialized Agents → Handle specific tasks, such as technical support or billing inquiries.

The Manager Agent is the key decision-maker in our workflow, which will receive the project title and invoke other agents for their respective tasks. Once each agent has completed their research, it will prepare a report on the research.

The Manager Agent is defined below:

notebook.ipynb

```
from crewai import Agent, LLM

llm = LLM(model="gpt-4o")

# Define the Manager AI
manager_agent = Agent(
    role="Project Research Manager",
    goal="Oversee the project research",
    backstory="""Your are an experienced project manager responsible
                 for ensuring project research.""",
    allow_delegation=True,
    verbose=True,
    llm=llm
)
```

Allows the manager to assign tasks

In the above code, `allow_delegation=True` lets the manager assign tasks, rather than performing them itself.

Next, the Market Demand Analyst AI is responsible for evaluating the demand for the proposed project. This involves analyzing industry trends, identifying potential users, and assessing competition. This is implemented below:

notebook.ipynb

```
# Define the Market Demang Analysis Agent
market_demand_agent = Agent(
    role="Market Demand Analyst",
    goal="Analyze market demand for new projects.",
    backstory="""A skilled market analyst with expertise
                 in evaluating product-market fit.""",
    allow_delegation=False,
    verbose=True,
    llm=llm
)
```

Cannot assign tasks to others

For this agent, we have `allow_delegation=False` because this Agent can only execute tasks—it cannot delegate them.

Moving on, the Risk Analysis Analyst AI focuses on identifying potential risks associated with the project. This includes technical, financial, and operational risks.

notebook.ipynb

```
# Define the Billing Support AI
risk_analysis_agent = Agent(
    role="Risk Analysis Analyst",
    goal="Assess potential risks associated with the project.",
    backstory="""A financial and strategic expert
                 focused on identifying business risks.""",
    allow_delegation=False,
    verbose=True,
    llm=llm
)
```

Cannot assign tasks to others

Finally, we have the Return on Investment (ROI) Analyst AI estimates the financial viability of the project. It evaluates projected costs, potential revenue, and expected returns over time.

notebook.ipynb

```
# Define the Billing Support AI
return_on_investment_agent = Agent(
    role="Return on Investment Analyst",
    goal="Estimate the financial return on investment.",
    backstory="""You are an expert in financial modeling
                 and investment analysis.""",
    allow_delegation=False,
    verbose=True,
    llm=llm
)
```

Cannot assign tasks to others

Next, we move onto defining their tasks.

Each research task is assigned to a specific AI agent to ensure structured execution. The Project Research Manager AI initiates and finalizes the research, while the specialized analysts conduct individual research tasks.

So starting with the first tasks, we have one for the Manager, which ensures that the research process is executed efficiently and that high-quality responses are generated.

notebook.ipynb

```
from crewai import Task

manager_task = Task(
    description="""Oversee the project research on {project_title}
                   and ensure timely, high-quality responses.""",
    expected_output="""A manager-approved response ready to be
                       sent as an article on {project_title}.""",
    agent=manager_agent,
)
```

Task for manager agent

The above task initiates the research workflow.

Moving on, the Market Demand Analyst AI analyzes whether there is a market need for the proposed project. Its Task is defined below:

notebook.ipynb

```
market_demand_task = Task(
    description="Analyze the demand for the project '{project_title}'.",

    expected_output="A structured summary of market demand trends.",

    agent=market_demand_agent,
)
```

Task for market demand agent

Next, the Risk Analysis Analyst AI identifies potential risks in the project. Its Task is defined below:

notebook.ipynb

```
risk_analysis_task = Task(
    description="Analyze the risk of the project title '{project_title}'.",

    expected_output="A categorized risk assessment report.",

    agent=risk_analysis_agent,
)
```

Task for
Risk analyst agent

The Return on Investment Analyst AI calculates whether the project will be financially viable. Its Task is defined below:

notebook.ipynb

```
return_on_investment_task = Task(
    description="Analyze the ROI of the project title '{project_title}'.",

    expected_output="A structured ROI estimate for the project.",

    agent=return_on_investment_agent,
)
```

Task for
ROI agent

Once all research tasks are completed, the Manager AI compiles the findings into a structured research report. So we define another task for it as follows:

notebook.ipynb

```
final_report_task = Task(
    description="""Review the final responses from the market demand,
                   risk analysis, and ROI agents and create a final report.""",
    expected_output="""A comprehensive report on the project '{project_title}'
                       containing the market demand, risk analysis,
                       and return on investment.""",
    agent=manager_agent,
)
```

Task for Manager agent

Moving on, we configure a Hierarchical Crew:

notebook.ipynb

```
from crewai import Crew, Process

project_research_crew = Crew(
    agents=[market_demand_agent, risk_analysis_agent, return_on_investment_agent],
    tasks=[
        market_demand_task,
        risk_analysis_task,
        return_on_investment_task,
        final_report_task
    ],
    manager_agent=manager_agent,

    process=Process.hierarchical,

    verbose=True,
)
```

Don't specify Manager agent here

Specify all tasks here

Specify the manager agent

Enable hierarchical execution

As discussed earlier, in a hierarchical workflow, we have two types of agents:

- The Manager Agent → Oversees the entire workflow, delegates tasks, and reviews outputs.
- Specialized Agents → Handle specific tasks, such as technical support or billing inquiries.

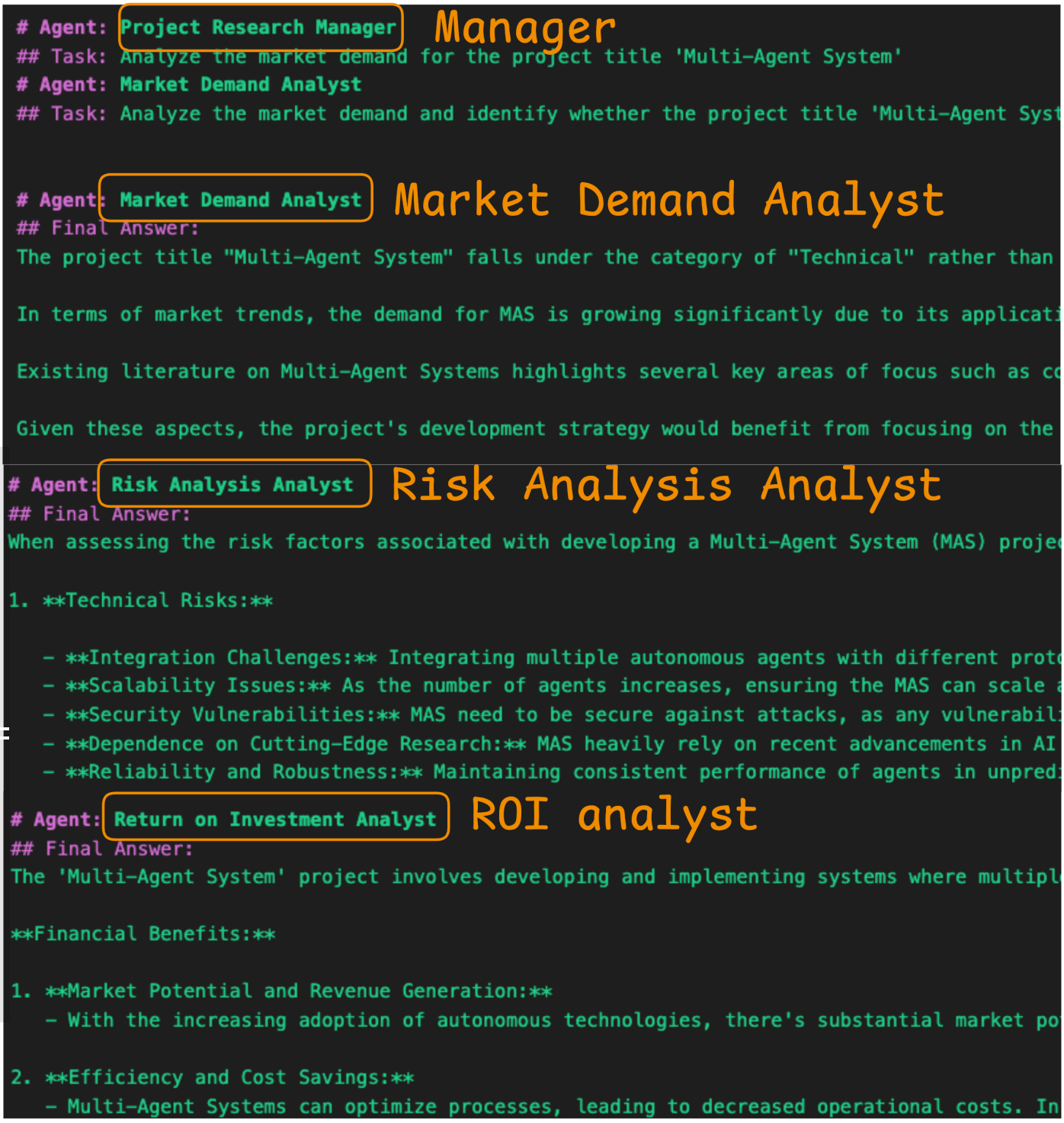
By specifying the `manager_agent` parameter, we specify the Agent that will act as the manager. All other agents are specified like we do usually.

Now that our Crew is assembled, we simulate a research by specifying a project title, which triggers the hierarchical AI workflow.

notebook.ipynb

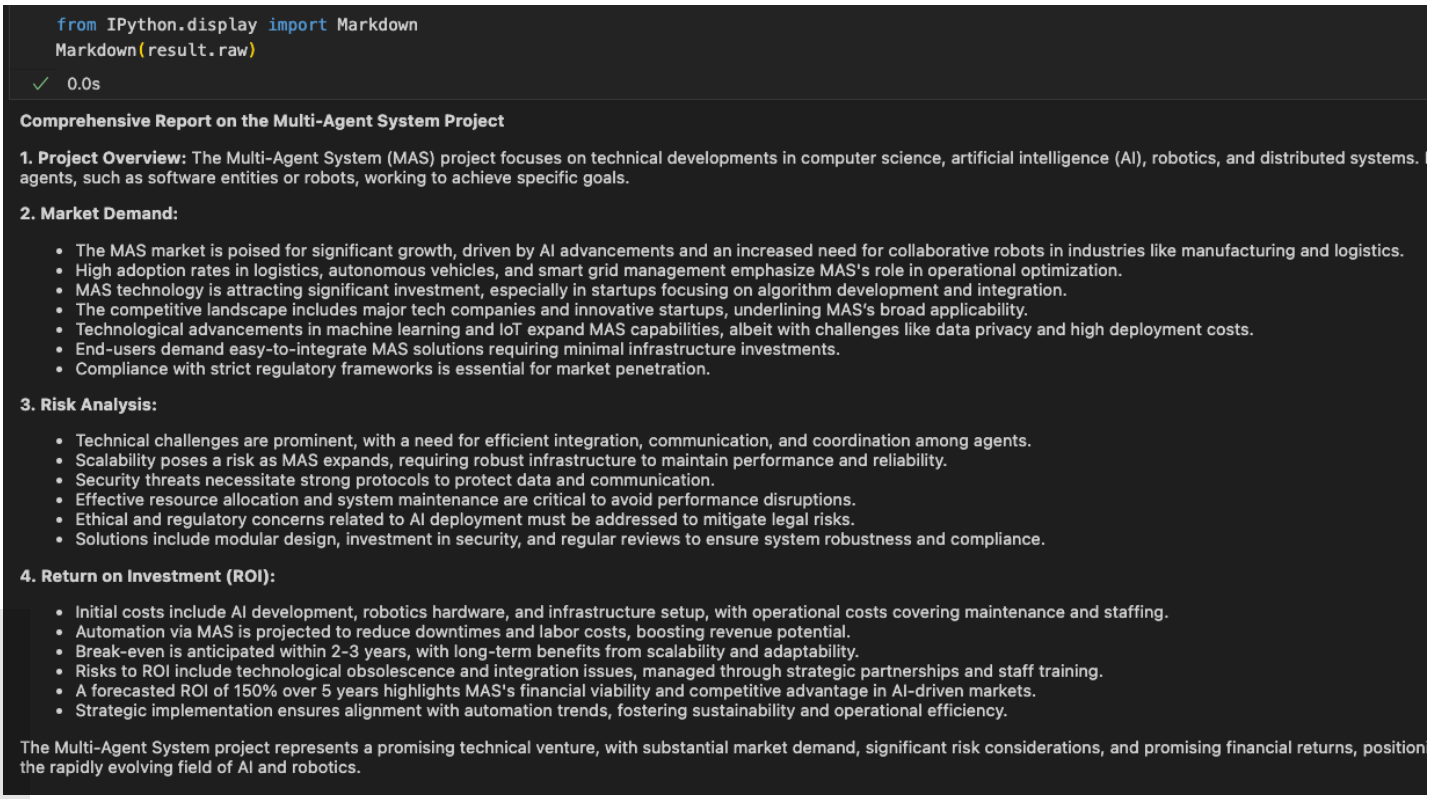
```
inputs = {"project_title": "Multi-Agent System"}

result = project_research_crew.kickoff(inputs=inputs)
```

Once we run this, we notice that at different stages of the execution, all Agents are executed:




```
# Agent: Project Research Manager
## Task: Analyze the market demand for the project title 'Multi-Agent System'
# Agent: Market Demand Analyst
## Task: Analyze the market demand and identify whether the project title 'Multi-Agent Syst


# Agent: Market Demand Analyst
## Final Answer:
The project title "Multi-Agent System" falls under the category of "Technical" rather than

In terms of market trends, the demand for MAS is growing significantly due to its applicati

Existing literature on Multi-Agent Systems highlights several key areas of focus such as co

Given these aspects, the project's development strategy would benefit from focusing on the

# Agent: Risk Analysis Analyst
## Final Answer:
When assessing the risk factors associated with developing a Multi-Agent System (MAS) projec

1. **Technical Risks:**

   - **Integration Challenges:** Integrating multiple autonomous agents with different proto
   - **Scalability Issues:** As the number of agents increases, ensuring the MAS can scale a
   - **Security Vulnerabilities:** MAS need to be secure against attacks, as any vulnerabili
   - **Dependence on Cutting-Edge Research:** MAS heavily rely on recent advancements in AI
   - **Reliability and Robustness:** Maintaining consistent performance of agents in unpredi

# Agent: Return on Investment Analyst
## Final Answer:
The 'Multi-Agent System' project involves developing and implementing systems where multipl

**Financial Benefits:**

1. **Market Potential and Revenue Generation:**
   - With the increasing adoption of autonomous technologies, there's substantial market po

2. **Efficiency and Cost Savings:**
   - Multi-Agent Systems can optimize processes, leading to decreased operational costs. In
```

And finally, we get the following response from the workflow, which covers exactly what we were looking for:



This is how Hierarchical processes work in case of Agentic systems.

> 💡 At this stage, we have a hands-on exercise for you. The current workflow does not leverage real-time online research—it relies solely on the AI agents' knowledge. To improve this, you have to integrate Serper Dev to allow our agents to pull live data from the web.
>
> This is what you need to do:
> - Get the Serper API Key and add it to your .env file.
> - Import the Serper Dev Tool from `crewai_tools` library and create a Serper Dev Tool object.
> - Add this to the Agents.
> - Run the workflow again.
>
> Let us know if you face any issues with it.

Human-in-the-loop Agents

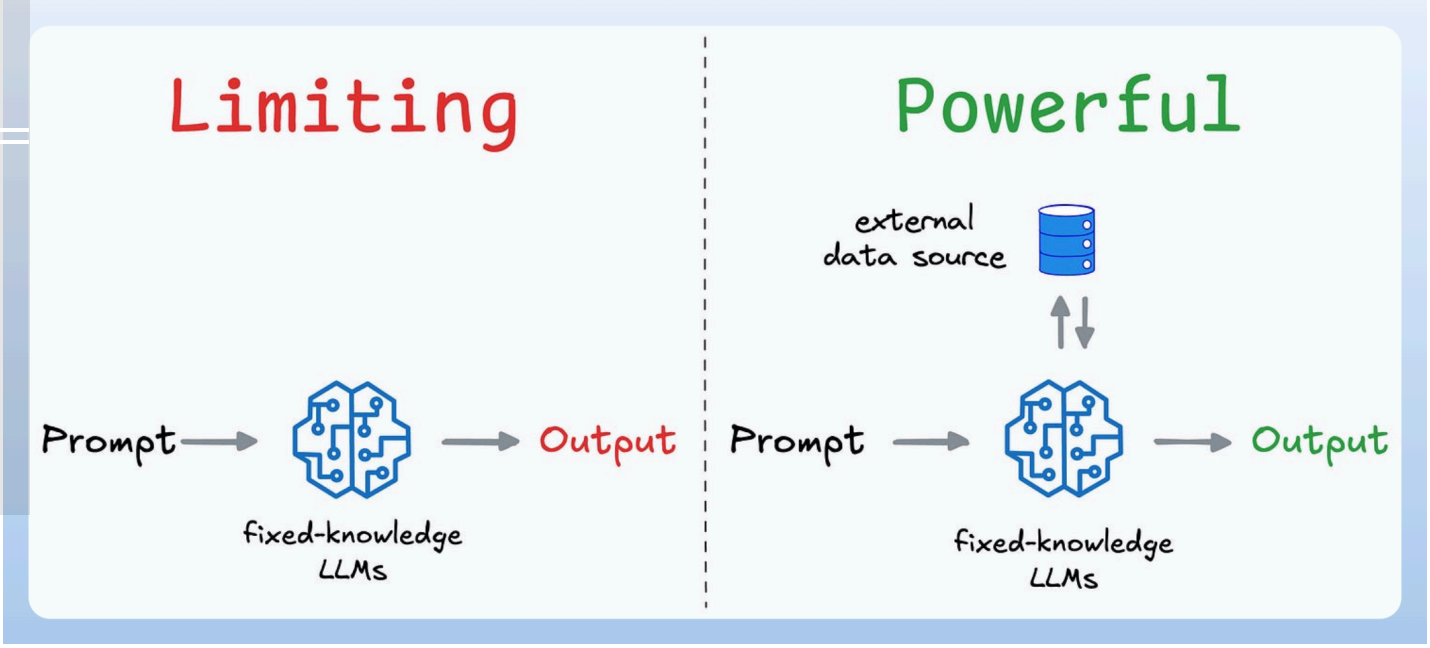
Before we get into building Human-in-the-loop Agents, let's dive in to understand a bit more about why they are important.

Background

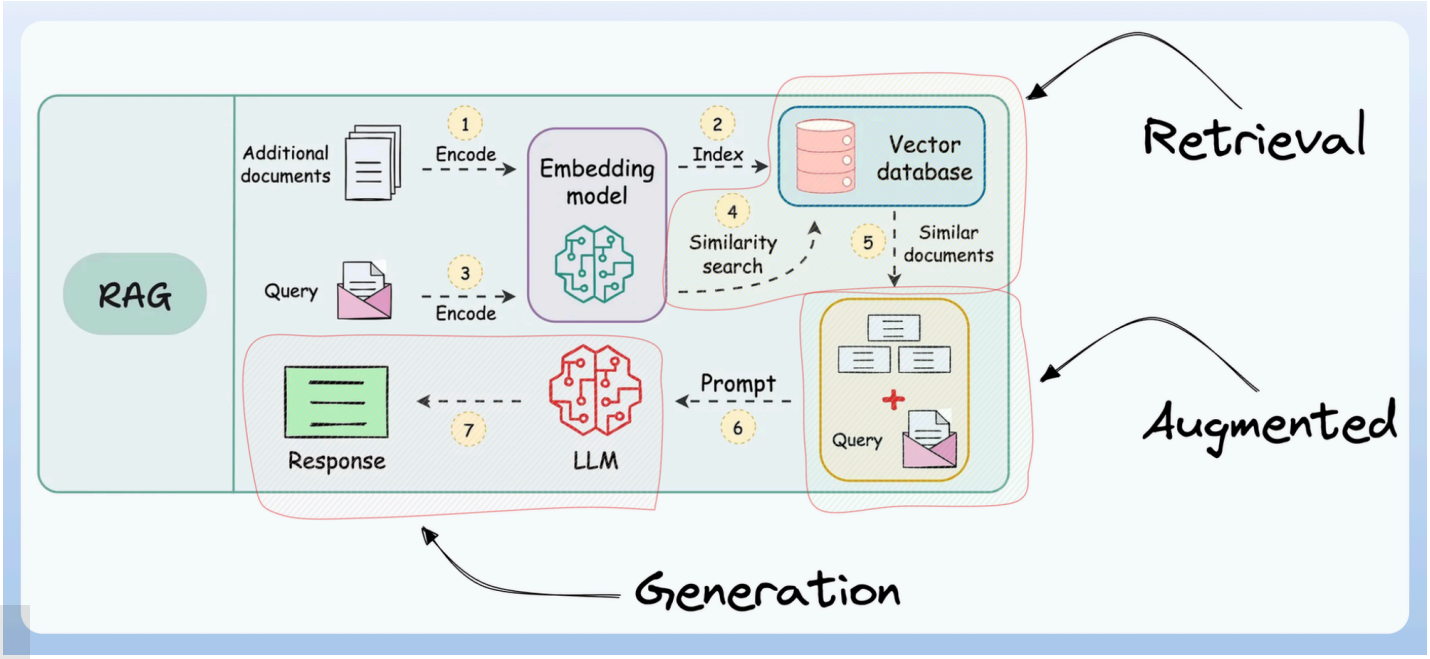
Given the scale and capabilities of modern LLMs, it feels limiting to use them as “standalone generative models” for pretty ordinary tasks like text summarization, text completion, code completion, etc.

Instead, their true potential is only realized when you build systems around these models, where they are allowed to:

- access, retrieve, and filter data from relevant sources,
- analyze and process this data to make real-time decisions and more.



RAG was a pretty successful step towards building such compound AI systems:



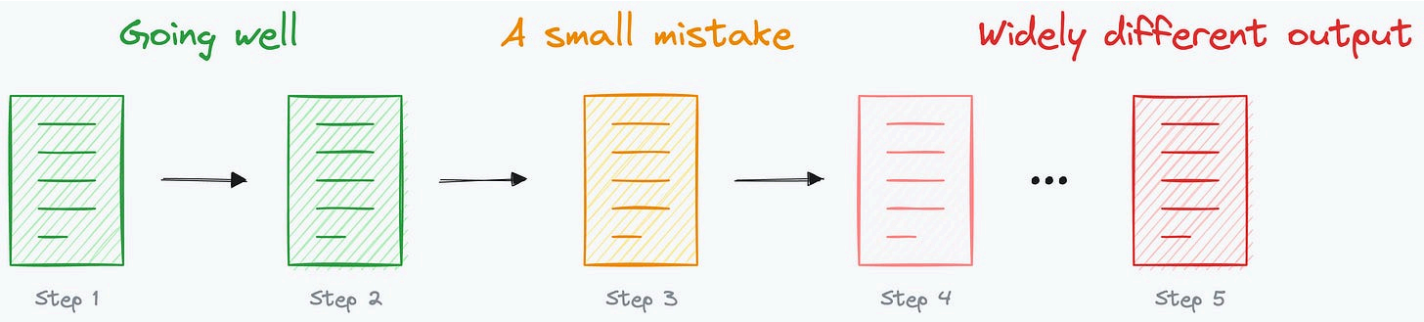
But since most RAG systems follow a programmatic flow (you, as a programmer, define the steps, the database to search for, the context to retrieve, etc.), it does not unlock the full autonomy one may expect these compound AI systems to possess.

That is why the primary focus in 2024 was (and going ahead in 2025 will be) on building and shipping **AI Agents**—autonomous systems that can reason, think, plan, figure out the relevant sources and extract information from them when needed, take actions, and even correct themselves if something goes wrong.

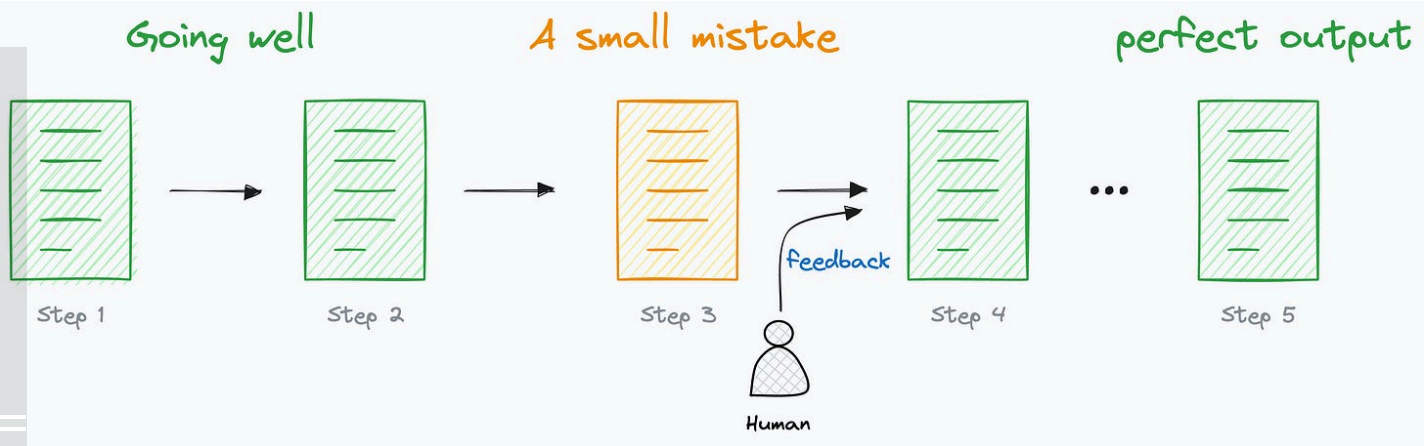
While this sounds promising, in their current state, the issue is that:

> ***We are far from building fully autonomous AI agents.***

More specifically, when an agent needs to perform a complex task with multiple steps, one mistake in the process derails the entire operation.



To avoid this, they need feedback mechanisms, such as human-in-the-loop (HITL), to guide them through the steps.



As the name suggests, human-in-the-loop workflows combine the power of AI Agents with humans.

This is important in several Agentic scenarios since it allows the Agents to collect more details or clarification from the human user as and when needed.

Technical details

Below, let's build a human-in-the-loop AI research system, where:

- An AI Researcher gathers and summarizes key industry insights.\
- A human reviewer checks and validates the AI-generated research.
- An AI Content Strategist writes a blog post based on the reviewed research.
- A human reviewer ensures the content is well-structured before publishing.

To begin, we define two AI agents:

- A Research Analyst → Uncovers insights about emerging AI trends.
- A Content Strategist → Converts research findings into engaging content.

notebook.ipynb

```
from crewai import Agent, LLM

llm = LLM(model="gpt-4o")

researcher_agent = Agent(
    role="Senior AI Researcher",
    goal="""Discover and summarize the latest
            trends in AI and technology.""",
    backstory="""An expert in AI research who tracks
                 emerging trends and their real-world applications.""",
    verbose=True,
    allow_delegation=False,
    llm=llm
)

content_strategist_agent = Agent(
    role="Tech Content Strategist",
    goal="""Transform AI research insights
            into compelling blog content.""",
    backstory="""An experienced tech writer who makes
                 AI advancements accessible to a broad audience.""",
    verbose=True,
    allow_delegation=True,
    llm=llm
)
```

To enable human validation, we set `human_input=True` during task definition in CrewAI, as demonstrated below:

notebook.ipynb

```
from crewai import Task

ai_research_task = Task(
    description=(
            "Conduct a deep analysis of AI trends in 2025. "
            "Identify key innovations, breakthroughs, and market shifts. "
            "Before finalizing, ask a human reviewer "
            "for feedback to refine the report."
    ),
    expected_output="""A structured research summary
                       covering AI advancements in 2025.""",
    agent=researcher_agent,
    human_input=True                    Require human input
)

blog_post_task = Task(
    description=(
        "Using insights from the AI Researcher, create an "
        " engaging blog post summarizing key AI advancements. "
        "Ensure the post is informative and accessible. Before "
        "finalizing, ask a human reviewer for approval."
    ),
    expected_output="A well-structured blog post on AI trends in 2025.",
    agent=content_strategist_agent,
    human_input=True                    Require human input
)
```

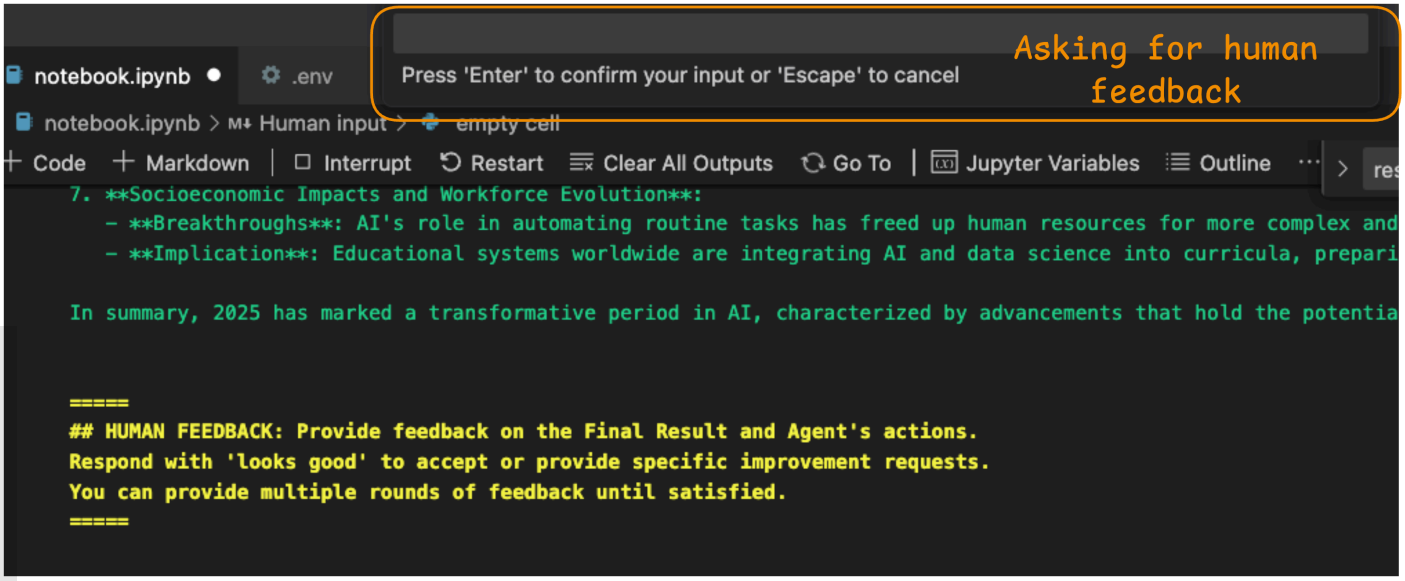
Now, we create a Crew:

notebook.ipynb

```
# Execute the workflow
result = ai_research_crew.kickoff()

# Display the final validated research output
print("\n=== Final AI Research Report ===")
print(result.raw)
```

Done!

Now, wherever we have asked the Agent to get a human input, it will prompt us for it, get some feedback (in natural language), rework the step if needed, and move to the next step only when the human approves it.



I typed "looks perfect" in the above human input field, which made it move to the next step, where it again asks for the input:

Finally, after all human-based evaluation, we get a well crafted response, which is expected to be much better than the one we would have obtained without any human feedback:

```
from IPython.display import Markdown
Markdown(result.raw)
```

✓ 0.0s

Title: The AI Frontier: Unveiling the Groundbreaking Trends of 2025

As we step into 2025, the artificial intelligence (AI) landscape has undergone a monumental transformation. A deluge of innovations is not only re
post delves into the cutting-edge advancements that are paving the way for an AI-driven future.

1. Neuromorphic Computing: The Brain-Inspired Revolution

Neuromorphic computing stands at the forefront of AI advancement, revolutionizing how machines process information by mimicking the human
become central to real-time, edge AI applications, significantly curtailing the energy demands of machine learning operations. They're powering
adept at swiftly processing sensory data to boost accuracy and response times. Imagine autonomous vehicles and robotic surgeons making split
of neuromorphic technology.

2. Quantum AI: Unlocking the Improbable

Quantum AI merges the capabilities of quantum computing with traditional AI algorithms, catalyzing unprecedented problem-solving capabilities.
discovery, where quantum algorithms facilitate intricate simulations of molecular interactions, and climate modeling, whereby they optimize renew
these technologies to streamline logistics and financial forecasts, fundamentally altering approaches to complex optimization challenges.

3. Generative AI: The New Canvas of Creativity

2025 marks a significant era for creative endeavors, with AI models generating art, music, and literature of astounding sophistication. Generative
innovative business opportunities and sparking debates on intellectual property and AI's role as a creative partner. This technology is not just aut
empowering artists with new tools and inspiring unprecedented forms of digital creativity.

4. AI-Driven Healthcare: A New Era of Precision and Personalization

Artificial intelligence is progressively reshaping healthcare, evidenced by its startling advancements in predictive diagnostics and personalized m
for more precise diagnoses than ever before, often surpassing human expertise. Integrating genetic profiling and treatment efficacy data, AI platf
outcomes. Additionally, continuous monitoring via AI ensures that chronic conditions are managed effectively, alerting healthcare providers to po
healthcare accessibility and quality.

5. Ethical Governance in AI: Building Trust and Accountability

The fast-paced evolution of AI underscores the urgency for ethical governance frameworks. 2025 has seen the establishment of robust policies e
Global regulations now aim to shield user privacy and mitigate algorithmic bias, fostering a culture of responsible innovation. Organizations world
and sociologists—to design AI systems that align with societal values and ethical norms.

6. Autonomous Systems: Pioneering the Future of Automation

Advancements in AI algorithms are fine-tuning autonomous systems, elevating their reliability and safety in real-world applications. By 2025, thes
seamlessly to dynamic environments. The demand for smart infrastructure is at an all-time high, driven by the necessity to support these autonor
5G technologies.

7. Socioeconomic Shifts: Redefining Work and Education

The impact of AI on socioeconomic structures is profound, as automation liberates human resources from mundane tasks, directing focus toward
institutions proactively embed AI and data science into curricula, equipping future generations with skills essential for AI-enhanced professions.
competencies towards interdisciplinary and technical expertise.

In conclusion, the AI trends of 2025 illustrate a milieu where technology not only enhances quality of life but also poses intricate challenges dema
drive economic and social progress, balanced innovation and vigilant governance are paramount in actualizing its full potential.

As we navigate these transformations, it's vital not just to embrace the technological advancements but to continuously evaluate their broader im
promise; it is a clarion call for responsible stewardship of technology towards a future of shared progress.

Done!

And that is how you can integrate human inputs into your existing Agentic workflows.

Multimodal Agents

Most AI systems are designed to process text, but many real-world applications require analyzing both text and images.

The good thing is that CrewAI also supports multimodal agents, allowing them to interpret and extract insights from visual content in addition to text.

As you can imagine, with multimodal capabilities, AI agents can:

- Process images from URLs or local paths.
- Analyze product quality, medical scans, or design elements.
- Integrate image and text-based reasoning for better decision-making.

Below, let's build a product quality inspector using a multimodal CrewAI agent, that works as follows:

1. The AI Agent reviews product images and identifies issues.
2. It detects defects such as scratches, dents, or misalignment.
3. It generates a structured quality assessment report.

To enable multimodal capabilities, we simply set `multimodal=True` when defining the agent.

notebook.ipynb

```python
from crewai import Agent, LLM

llm = LLM(model="gpt-4o")

quality_inspector = Agent(
    role="Product Quality Inspector",
    goal="Analyze and assess the quality of product images",
    backstory="""An experienced manufacturing quality control
                 expert who specializes in detecting defects
                 and ensuring compliance.""",
    multimodal=True,
    verbose=True,
    llm=llm
)
```

Enable multimodal capabilities

When we set `multimodal=True`:

- The agent automatically includes image analysis capabilities.
- It can process and interpret visual data without additional configuration.
- It uses built-in tools to analyze images efficiently.

> 💡 Under the hood, one of the key tools CrewAI uses to process and analyze images is the `AddImageTool`.
>
> This tool enables multimodal agents to retrieve images from URLs or local file paths, process image data efficiently with pre-configured AI capabilities and analyze visual content based on task-specific instructions.

Moving on, the AI agent needs a task description that tells it what aspects of the image to analyze.

notebook.ipynb

Image path or url

```
from crewai import Task

# Define a Task for Product Image Inspection
inspection_task = Task(
    description="""Inspect the product image at {image_url}.
                   Identify any visible defects such as scratches,
                   dents, misalignment, or color inconsistencies.
                   Provide a structured quality assessment report.""",
    expected_output="""A detailed report highlighting detected
                       issues and overall quality score.""",
    agent=quality_inspector
)
```

When creating Task descriptions:

- Be specific about what the agent should analyze.
- Provide context on key issues to look for.
- Ensure the image URL is accessible before running the task.

Now that we have a multimodal AI agent and a task, we can create a Crew and execute the workflow.

notebook.ipynb

```
from crewai import Crew

# Create a Crew with the Multimodal Agent
quality_inspection_crew = Crew(
    agents=[quality_inspector],
    tasks=[inspection_task],
    verbose=True
)

image_url = "https://s.marketwatch.com/public/resources/images/MW-HT101_nerd_d_ZG_20191010165334.jpg"

# Run the workflow
result = quality_inspection_crew.kickoff(inputs={"image_url": image_url})

# Display the final inspection report
print("\n=== Final Product Quality Report ===")
print(result.raw)
```

In the above code:

- The Crew executes the inspection task.
- The multimodal agent retrieves and analyzes the product image.
- It generates a structured quality report based on detected issues.

This is the image specified above:

Based on this, here's the output we get from the above execution:

```
Overriding of current TracerProvider is not allowed
# Agent: Product Quality Inspector
## Task:
    Inspect the product image at https://s.marketwatch.com/public/resources/images/MW-HT101_nerd_d_ZG_20191010165334.jpg.
    Identify any visible defects such as scratches, dents, misalignment, or color inconsistencies.
    Provide a structured quality assessment report.



# Agent: Product Quality Inspector
## Final Answer:
Quality Assessment Report:

1. **Visible Defects Detected:**
   - **Dents and Creasing:** The box exhibits significant denting and creases, suggesting improper handling or storage.
   - **Tape Misalignment:** The tape used on the box is not aligned properly, indicating a hasty or careless packaging process.
   - **Edge Damages:** The edges of the box are damaged, which can compromise the integrity of the contents.

2. **Overall Quality Score:**
   - **Score:** 3/10
   - **Remarks:** The packaging is severely compromised and does not meet standard quality benchmarks. Action is required for improvement in pa

Recommendations include using sturdier packaging materials and ensuring proper handling and alignment of sealing tapes to enhance product prote
```

```
=== Final Product Quality Report ===
Quality Assessment Report:

1. **Visible Defects Detected:**
   - **Dents and Creasing:** The box exhibits significant denting and creases, suggesting improper handling or storage.
   - **Tape Misalignment:** The tape used on the box is not aligned properly, indicating a hasty or careless packaging process.
   - **Edge Damages:** The edges of the box are damaged, which can compromise the integrity of the contents.

2. **Overall Quality Score:**
   - **Score:** 3/10
   - **Remarks:** The packaging is severely compromised and does not meet standard quality benchmarks. Action is required for improvement in pa

Recommendations include using sturdier packaging materials and ensuring proper handling and alignment of sealing tapes to enhance product prote
```

The Quality Assessment Report says:

#1) Visible Defects Detected:

- *Dents and Creasing: The box exhibits significant denting and creases, suggesting improper handling or storage.*
- *Tape Misalignment: The tape used on the box is not aligned properly, indicating a hasty or careless packaging process.*
- *Edge Damages: The edges of the box are damaged, which can compromise the integrity of the contents.*

#2) Overall Quality Score:

- *Score: 3/10*
- *Remarks: The packaging is severely compromised and does not meet standard quality benchmarks. Action is required for improvement in*

packaging processes to prevent damage during transit.

Recommendations include using sturdier packaging materials and ensuring proper handling and alignment of sealing tapes to enhance product protection.

All of this is indeed correct based on the image we have above, which shows that the Agent was able to interpret the image provided above.

That said, building multimodal Agents can be a bit tricky at times, so make sure to remember these points when building them:

- Provide direct URLs to images or use absolute file paths.
- If using local images, consider hosting them temporarily for accessibility.
- Be specific about what to analyze in the image. Include clear questions to guide AI interpretation.
- Image processing requires more computational power than text analysis. Consider compressing images to make them smaller.
- Ensure your LLM supports multimodal processing.
- Have fallback mechanisms if image processing fails. Use logging to monitor AI performance on different image types.

Conclusion, takeaways, and next steps

With that, we come to the end of Part 6 of the Agents crash course.

In this part we expanded the capabilities of AI agents by introducing techniques that allow AI to work more dynamically with humans, operate in structured hierarchies, and process multimodal inputs.

Here are the key takeaways from Part 6:

- Integrating Human-in-the-Loop AI

    - We saw how to add human oversight into agentic workflows using `human_input=True`.
    - This allows users to review, refine, or approve AI outputs before finalization.
    - Human-in-the-loop AI is essential for critical decision-making workflows such as medical AI, financial reports, or content creation.
- Hierarchical Processes for AI Workflows
    - Instead of executing tasks in a linear sequence, we introduced manager agents to oversee task delegation.
    - Hierarchical AI structures mimic real-world organizations, where tasks are divided among specialized agents with a manager ensuring quality and consistency.
    - This structure makes AI more scalable, as new agents can be added without disrupting workflow efficiency.
- Building Multimodal Agents
    - AI systems are not limited to text—we enabled multimodal capabilities so agents could process both text and images.
    - By integrating image analysis tools, agents could inspect product quality, analyze medical scans, and extract insights from visual content.
    - This bridges the gap between language-based AI and vision-based AI, making automation more useful in practical scenarios.
- Understanding the Tools Behind Multimodal Processing
    - CrewAI automatically equips multimodal agents with tools like AddImageTool, allowing them to retrieve and analyze image data.
    - We discussed how AI can use structured schemas to process visual inputs alongside text for enhanced contextual understanding.

💡 As a key takeaway, always think of your agentic system as a structured design—just as you would in a real-world human workflow. When designing an AI-driven system, ask yourself:

- How would I structure this process if I were working with a team of humans?

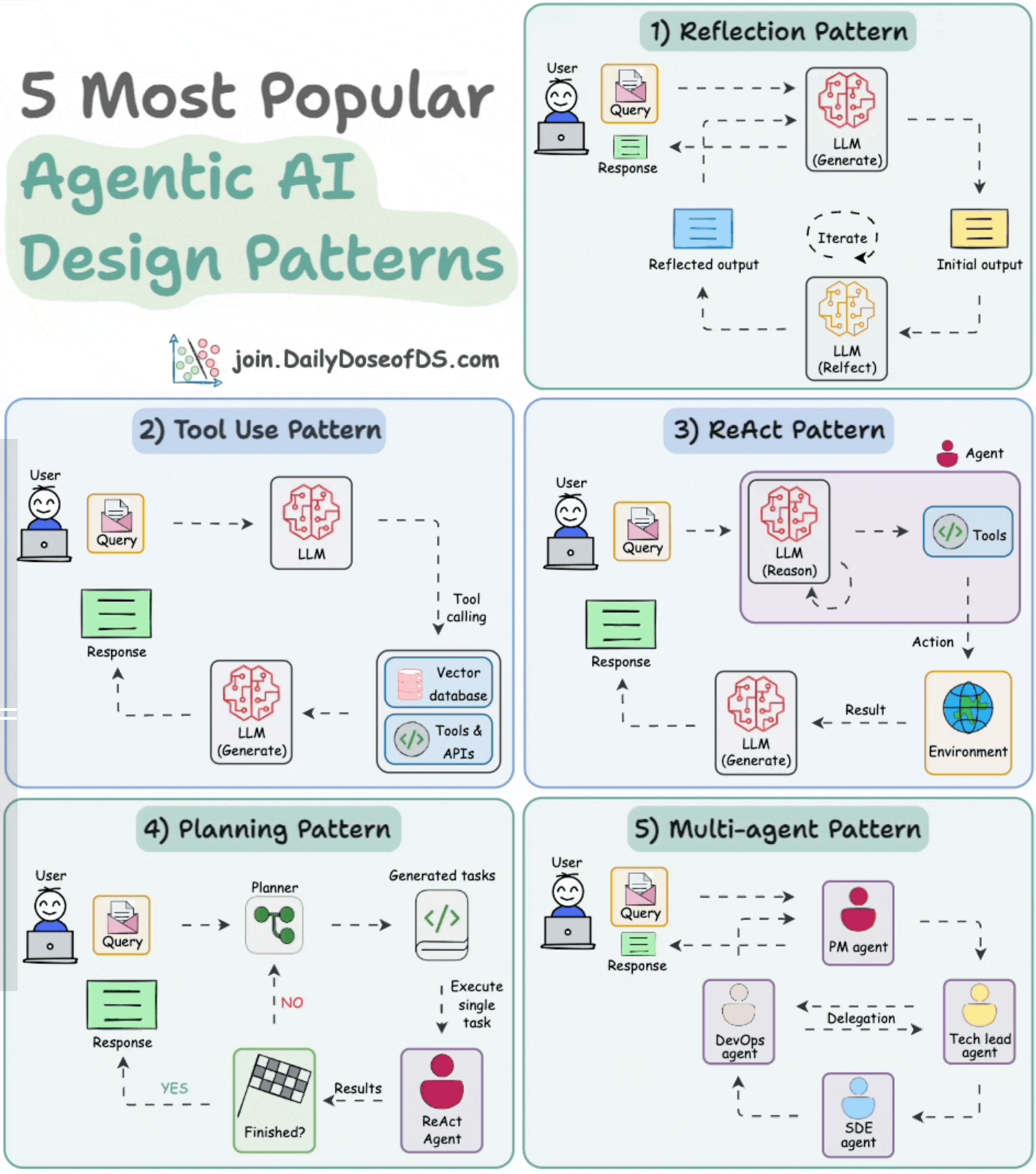
- Which tasks need to be done first, and which can be done in parallel?
- Who (or which AI agent) should oversee and validate the work?
- Which Agents should be enforced with guardrail?
- and more.

Once you have this layout on paper, it becomes immensely easier to extend this to an implementation.

That said, note that we aren't done yet.

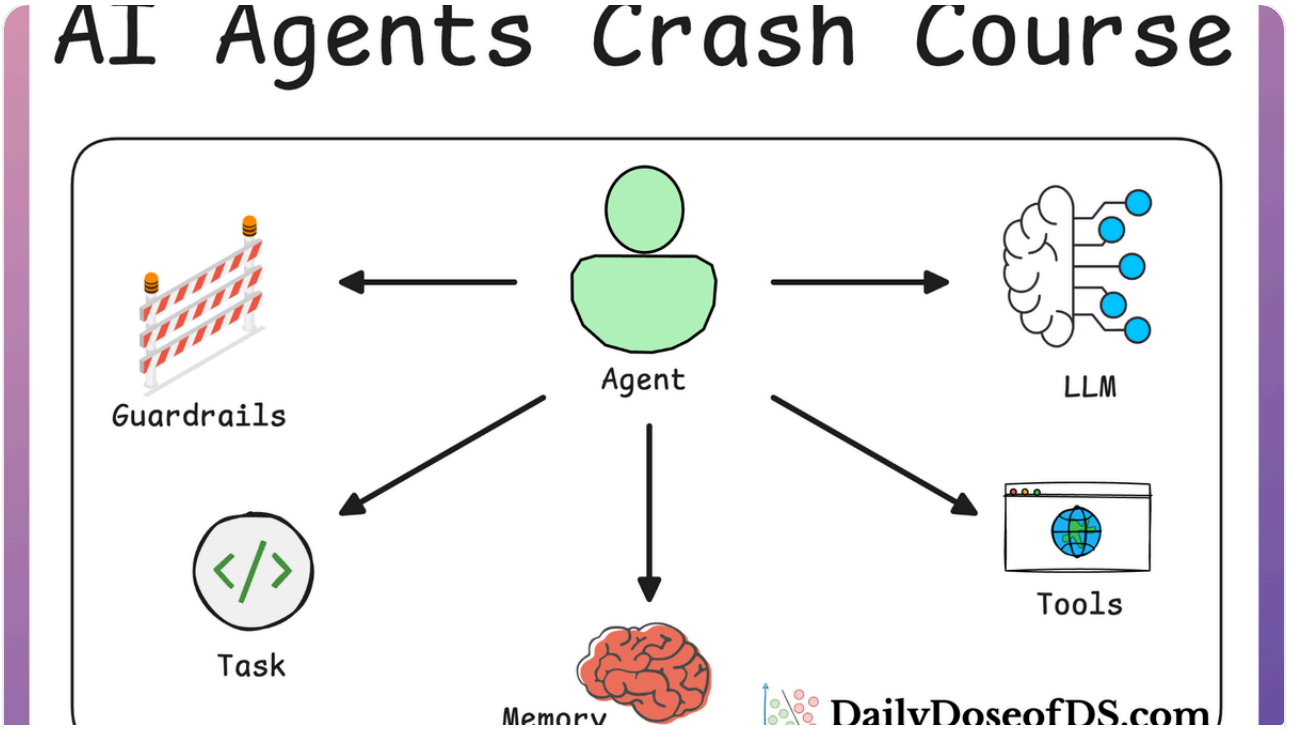
In the upcoming parts, we have several other advanced agentic things planned for you:

- Building production-ready agentic pipelines that scale.
- Agentic RAG (Retrieval-Augmented Generation) – combining RAG with AI agents.
- Optimizing agents for real-world applications in business and automation.
- Building Agents around the Agentic patterns depicted below:



- and many many more.

Read the next part of this crash course here:



A Practical Deep Dive Into Knowledge for Agentic Systems

AI Agents Crash Course—Part 7 (with implementation).

Daily Dose of Data Science • Avi Chawla

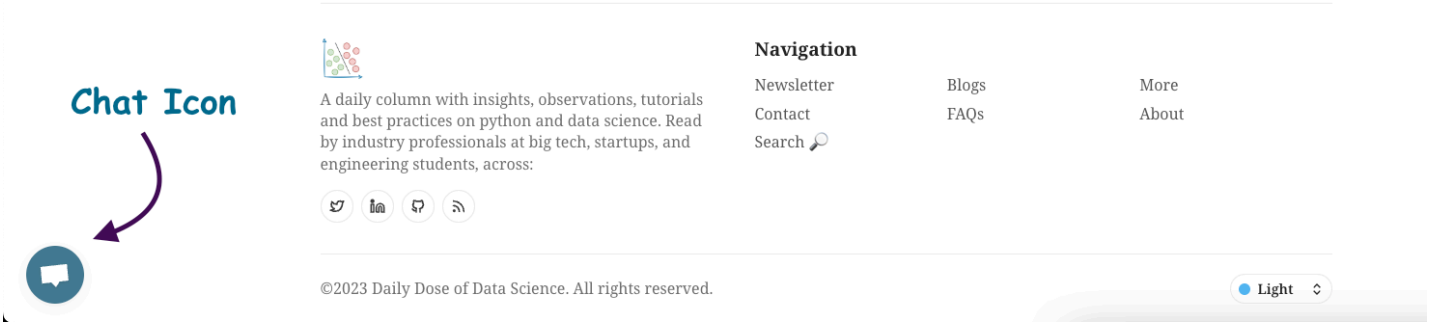
As always, thanks for reading!

Any questions?

Feel free to post them in the comments.

Or

If you wish to connect privately, feel free to initiate a chat here:



Connect via chat

Agents LLMs AI Agent Crash Course

Share this article

Read next

MCP Jun 15, 2025 • 22 min read

The Full MCP Blueprint: Building a Full-Fledged MCP Workflow using Tools, Resources, and Prompts

Model context protocol crash course—Part 4.

Avi Chawla, Akshay Pachaar

Agents Jun 8, 2025 • 20 min read

The Full MCP Blueprint: Building a Custom MCP Client from Scratch

Model context protocol crash course—Part 3.

Avi Chawla, Akshay Pachaar

Agents Jun 1, 2025 • 22 min read

The Full MCP Blueprint: Background, Foundations, Architecture, and Practical Usage (Part B)

Model context protocol crash course—Part 2.

Avi Chawla, Akshay Pachaar

A daily column with insights, observations, tutorials and best practices on python and data science. Read by industry professionals at big tech, startups, and engineering students, across:

Navigation

Sponsor Newsletter More Contact