In [1]:
```
!pip install crewai==0.28.8 crewai_tools==0.1.6 langchain_community==0.0.29
```

```
Requirement already satisfied: crewai==0.28.8 in c:\programdata\anaconda3\lib\site-packages (0.28.8)
Requirement already satisfied: crewai_tools==0.1.6 in c:\programdata\anaconda3\lib\site-packages (0.1.6)
Requirement already satisfied: langchain_community==0.0.29 in c:\programdata\anaconda3\lib\site-packages (0.0.29)
Requirement already satisfied: appdirs<2.0.0,>=1.4.4 in c:\programdata\anaconda3\lib\site-packages (from crewai==
0.28.8) (1.4.4)
Requirement already satisfied: click<9.0.0,>=8.1.7 in c:\programdata\anaconda3\lib\site-packages (from crewai==0.
28.8) (8.1.7)
Requirement already satisfied: embedchain<0.2.0,>=0.1.98 in c:\programdata\anaconda3\lib\site-packages (from crew
ai==0.28.8) (0.1.113)
Requirement already satisfied: instructor<0.6.0,>=0.5.2 in c:\programdata\anaconda3\lib\site-packages (from crewa
i==0.28.8) (0.5.2)
Requirement already satisfied: langchain<0.2.0,>=0.1.10 in c:\programdata\anaconda3\lib\site-packages (from crewa
i==0.28.8) (0.1.13)
Requirement already satisfied: openai<2.0.0,>=1.13.3 in c:\programdata\anaconda3\lib\site-packages (from crewai==
0.28.8) (1.52.2)
Requirement already satisfied: opentelemetry-api<2.0.0,>=1.22.0 in c:\programdata\anaconda3\lib\site-packages (fr
om crewai==0.28.8) (1.27.0)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-http<2.0.0,>=1.22.0 in c:\programdata\anaconda3
\lib\site-packages (from crewai==0.28.8) (1.27.0)
```

In [2]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
!pip uninstall backports
```

```
WARNING: Skipping backports as it is not installed.
```

In [4]:
```
!pip install backports.tarfile
```

```
Requirement already satisfied: backports.tarfile in c:\programdata\anaconda3\lib\site-packages (1.2.0)
```

In [5]:
```
from crewai import Agent, Task, Crew
from langchain_google_genai import ChatGoogleGenerativeAI
```

In [39]:
```
import os

# API Keys
gemini_api_key = ""
serper_api_key = ""

# Set environment variables
os.environ["GEMINI_API_KEY"] = gemini_api_key
os.environ["GEMINI_MODEL_NAME"] = 'gemini-1.5-flash'
os.environ["SERPER_API_KEY"] = serper_api_key

llm = ChatGoogleGenerativeAI(
    api_key=gemini_api_key,
    model=os.environ["GEMINI_MODEL_NAME"],
    google_api_key=gemini_api_key
)
```

In [40]:
```
from crewai_tools import ScrapeWebsiteTool, SerperDevTool

search_tool = SerperDevTool()
scrape_tool = ScrapeWebsiteTool()
```

In [41]:
```
research_agent = Agent(
    role="Industry Researcher",
    goal="Research the industry and the company,"
          "Provide a brief paragraph about the company and its insights, including key trends,market opportunities, a
    backstory="As a knowledgeable researcher, your mission is to dive deep into"
        "industry trends and uncover valuable insights that guide innovation.",
    verbose=True,
    allow_delegation=True,
    tools = [scrape_tool, search_tool],
    llm=llm
)
```

In [42]:
```
use_case_agent = Agent(
    role="Use Case Generator",
    goal="Analyze the research on the industry and the company,"
          "Based on the industry trends and the company\'s strategic focus areas, propose three relevant AI and ML us
          "Include use cases for GenAI, LLMs, or ML technologies that can enhance operations, customer experience, or
    backstory="You excel at creating AI and ML use cases that are tailored to the specific needs of the industry and
              "Your ability to translate industry trends into actionable use cases helps drive innovation and improve
    verbose=True,
    allow_delegation=True,
    tools = [scrape_tool, search_tool],
    llm=llm
)
```

In [44]:
```
resource_collector_agent = Agent(
    role="Resource Collector",
    goal="Gather one resource or dataset from each platform kaggle, github, and huggingface "
        "to support the research and use case generation tasks. Additionally, compile and print the outputs from "
        "the 'Industry Researcher' and 'Use Case Generator' agents for a final report.",
    backstory="This agent specializes in collecting data and resources that will aid in the analysis and development
        "of AI use cases. Additionally, it compiles the research and use case findings into a final output, ensuring
        "that all necessary information is at hand for effective decision-making.",
    allow_delegation=True,
    tools = [scrape_tool, search_tool],
    llm=llm
)
```

In [45]:
```
research_task = Task(
    description=(
        "Research the industry and the company, "
        "Provide a brief paragraph about the company and its insights, including key trends, "
        "market opportunities, and potential risks."
    ),
    expected_output=(
        "A concise summary of the company's position in the {topic} industry, "
        "highlighting key insights and trends for {company_name}."
    ),
    agent=research_agent,
)
```

In [46]:
```
use_case_task = Task(
    description=(
        "Analyze the research on the industry and the company, "
        "Based on the industry trends and the company's strategic focus areas, propose three relevant AI and ML use c
        "Include use cases for GenAI, LLMs, or ML technologies that can enhance operations, customer experience, or s
    ),
    expected_output=(
        "Three detailed use cases for AI and ML applications relevant to {company_name} in the {topic} industry."

    ),
    agent=use_case_agent,
)
```

In [48]:
```
resource_collection_task = Task(
    description=(
                "Gather one relevant resource or dataset from each kaggle, github, and huggingface "
        "to support the research and use case generation tasks. Also, compile and display the outputs from the "
        "'Industry Researcher' and 'Use Case Generator' agents."
    ),
    expected_output=(
        "A collection of datasets and resources pertinent to {company_name} and the {topic} industry. "
        "Also, a compiled output of the first two agents' findings."


    ),
    agent=resource_collector_agent,
)
```

In [49]:
```
from crewai import Crew, Process
```

In [50]:
```
from langchain_google_genai import ChatGoogleGenerativeAI
```

In [51]:
```
multi_inputs = {
    'topic': 'technology',
    'company_name': 'amazon'
}
```

In [36]:
```
multi = Crew(
    agents=[research_agent,
            use_case_agent,
            resource_collector_agent],

    tasks=[research_task,
           use_case_task,
           resource_collection_task],

    manager_llm=ChatGoogleGenerativeAI(
        model="gemini-1.5-flash",
        temperature=0.5,
        google_api_key=gemini_api_key
    ),
    process=Process.sequential,
    verbose=True
)
```

```
2024-10-25 11:04:18,244 - 42576 - __init__.py-__init__:538 - WARNING: Overriding of current TracerProvider is not al
lowed
```

In [38]:
```
from IPython.display import Markdown

result = multi.kickoff(inputs=multi_inputs)

# Ensure that result contains outputs from all tasks
if isinstance(result, dict):
    # Initialize a string to accumulate markdown-formatted outputs
    markdown_output = ""

    # Iterate through each task result and append it to the markdown string
    for task_name, task_output in result.items():
        markdown_output += f"### Output from {task_name}:\n\n{task_output}\n\n"

    # Display the concatenated markdown
    display(Markdown(markdown_output))

else:
    # If result is not a dict, handle it accordingly
    display(Markdown(f"### Output:\n\n{result}"))
```

```
 [DEBUG]: == Working Agent: Industry Researcher
 [INFO]: == Starting Task: Research the industry and the company, Provide a brief paragraph about the company and
its insights, including key trends, market opportunities, and potential risks.


> Entering new CrewAgentExecutor chain...
Thought: I need to gather information about Amazon to understand its position in the tech industry, key insights,
trends, opportunities, and risks.

Action: Read website content
Action Input: {"website_url": "https://www.amazon.com/"}


Amazon.com. Spend less. Smile more.
Skip to main content
.us
 Deliver to
 India
All
```

In [63]:
```
from IPython.display import Markdown
Markdown(result)
```

Out[63]:

- **Kaggle:** "Amazon Cell Phones Reviews" dataset. This dataset contains reviews for unlocked and locked carriers across ten brands, including Apple, Samsung, Google, and more. It can be used to understand customer sentiment, preferences, and buying behavior in the smartphone market, which is relevant to Amazon's technology business.
- **GitHub:** "aws/amazon-sagemaker-examples" repository. This repository contains a collection of Jupyter notebooks that demonstrate how to build, train, and deploy machine learning models using Amazon SageMaker. It covers a wide range of use cases, including personalized shopping experiences, supply chain optimization, and fraud detection.
- **Hugging Face:** The Amazon Web Services (AWS) organization page. This page showcases models and datasets specifically designed for use with AWS services. It includes models like Chronos for time series forecasting and datasets like AmazonQAC for question answering. This resource provides insights into Amazon's AI/ML initiatives and can help explore use cases. forecasting and datasets like AmazonQAC for question answering. This resource provides insights into Amazon's AI/ML initiatives and can help explore use cases.

In [53]:
```
result = multi.kickoff(inputs=multi_inputs)

# Print the result object to understand its structure
print(result)

# Now, assuming the result is a structured string or similar, we can print each agent's output manually.
# If result is a single string containing everything, you can print it directly:
if isinstance(result, str):
    # Directly print the combined result
    from IPython.display import Markdown
    display(Markdown(result))
else:
    # Handle structured result (if result turns out to be a dictionary or list)
    first_agent_output = result.get('research_task', 'No output from research agent')
    second_agent_output = result.get('use_case_task', 'No output from use case agent')
    third_agent_output = result.get('resource_collection_task', 'No output from resource collector agent')

    # Combine the outputs into a markdown-friendly format
    combined_output = f"""
    ### First Agent (Industry Researcher) Output:
    {first_agent_output}

    ### Second Agent (Use Case Generator) Output:
    {second_agent_output}

    ### Third Agent (Resource Collector) Output:
    {third_agent_output}
    """

    # Display the combined result
    from IPython.display import Markdown
    display(Markdown(combined_output))
```

```
 [DEBUG]: == Working Agent: Industry Researcher
 [INFO]: == Starting Task: Research the industry and the company, Provide a brief paragraph about the company and its insights, including key trends, market opportunities, and potential risks.


> Entering new CrewAgentExecutor chain...
Thought: I need to gather information about Amazon and the technology industry. I can start by reading Amazon's website to understand their business and mission.

Action: Read website content
Action Input: {"website_url": "https://www.amazon.com"}

Amazon.com. Spend less. Smile more.
Skip to main content
.us
 Deliver to
 India
All
Select the department you want to search in
```

In [ ]: