Title: Generating AI/GenAI Use Cases for Industry-Specific Applications

Objective

The aim is to create a **Multi-Agent System** that generates AI and Generative AI (GenAI) use cases for a specific company or industry. This system conducts market research, identifies relevant industries, and suggests resource assets for AI/ML solutions with a focus on operations, supply chain, and customer experience.

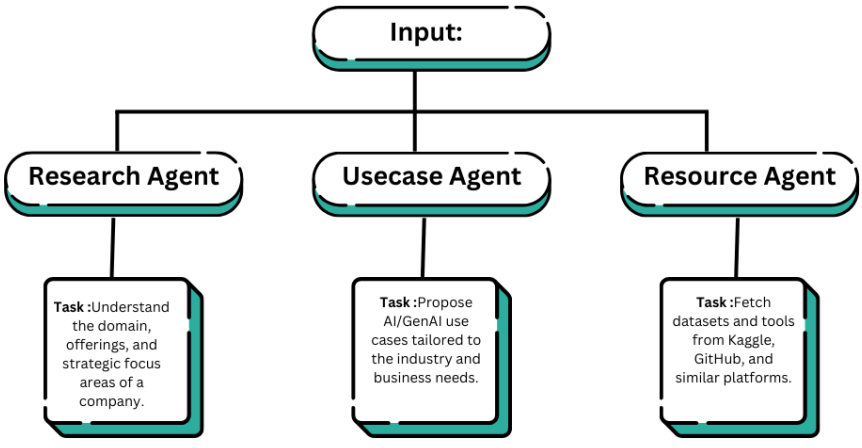
System Design

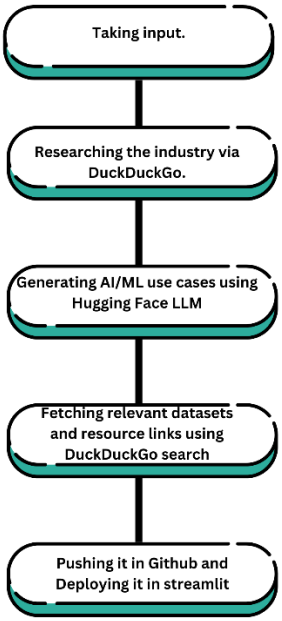
Overview

The system uses a Multi-Agent architecture with three key agents:

1. **Research Agent**:
   Conducts market research to infer the industry and gather insights.
   Tool: DuckDuckGo search.
2. **Use Case Agent**:
   Generates AI/ML use cases based on the inferred industry.
   Tool: Hugging Face LLM API.
3. **Resource Agent**:
   Searches for datasets and resource links relevant to the company and industry.
   Tool: DuckDuckGo search with targeted queries.

Flow Charts





Methodology

Input:

- **Company Name** (string).

Workflow:

1. **Research Agent**:
    - Uses the company name to query its industry using DuckDuckGo.
    - Extracts industry-specific keywords from search results.
2. **Use Case Agent**:
    - Sends the inferred industry to the Hugging Face LLM.
    - Generates actionable AI/ML use cases focusing on operations, supply chain, and customer experience.
3. **Resource Agent**:
    - Constructs queries for relevant datasets or resources on Kaggle and GitHub.
    - Formats the extracted links and resources.

4. **Output**:
    - Industry Insights
    - AI/ML Use Cases
    - Dataset and Resource Links

Code implementation :

```
import streamlit as st
from langchain_community.tools import DuckDuckGoSearchRun
from huggingface_hub import InferenceClient

# Initialize DuckDuckGoSearchRun and Hugging Face client
search = DuckDuckGoSearchRun()
client = InferenceClient(api_key="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")

# Streamlit App
st.title("AI Use Case Generator ")

# Input field for company name
company = st.text_input("Enter the Company Name:")

# Function to infer industry based on company name
def infer_industry(company):
    query = f"{company} industry"
    results = search.invoke(query)
    return results

# Function to generate AI/ML use cases based on the industry
def generate_use_cases_with_hf(industry):
    messages = [
        {
            "role": "user",
            "content": f"Suggest relevant AI and GenAI use cases for the {industry} industry, focusing on operations, supply chain, and customer experience. Highlight actionable insights and potential challenges."
        }
    ]
```

```
    response = ""
    with st.spinner("Generating use cases with Hugging Face LLM..."):
        stream = client.chat.completions.create(
            model="mistralai/Mistral-7B-Instruct-v0.3",
            messages=messages,
            max_tokens=500,
            stream=True
        )
        for chunk in stream:
            delta = chunk.choices[0].delta.content
            response += delta

    return response.strip()  # Clean up any leading/trailing whitespace

# Function to fetch relevant links (datasets or resources)
def fetch_relevant_links(company, industry):
    query = f"{company} {industry} datasets site:kaggle.com OR site:github.com"
    results = search.invoke(query)

    # Check if results are valid and format them
    if results:
        github_links = [result for result in results if 'github.com' in result]
        kaggle_links = [result for result in results if 'kaggle.com' in result]

        formatted_results = []
        if github_links:
            formatted_results.append(f"**GitHub Links:**")
            formatted_results.extend(github_links)

        if kaggle_links:
            formatted_results.append(f"**Kaggle Links:**")
            formatted_results.extend(kaggle_links)
```

```
        return "\n".join(formatted_results) if formatted_results else "No relevant datasets or resources found."
    else:
        return "No relevant datasets or resources found."


# Streamlit Workflow
if st.button("Generate") and company:
    with st.spinner("Processing..."):
        try:
            # Step 1: Industry Insights
            st.subheader("Industry Insights")
            industry_info = infer_industry(company)
            st.write(industry_info)

            # Step 2: AI Use Cases
            st.subheader("AI/ML Use Cases")
            use_cases = generate_use_cases_with_hf(industry_info)
            st.write(use_cases)

            # Step 3: Dataset and Resource Links
            st.subheader("Relevant Datasets and Resources")
            links = fetch_relevant_links(company, industry_info)
            st.write(links)

            st.success("Data Generated Successfully!")
        except Exception as e:
            st.error(f"An error occurred: {e}")
```

Requirements.txt

```
streamlit
langchain
llama-index
huggingface-hub
```

accelerate

langchain_community

duckduckgo-search

Outputs

Industry Insights: Provides a summary of the company's industry and strategic areas.

AI/ML Use Cases: Suggests actionable solutions tailored to the company's operations and goals.

Resource Links: Lists datasets and tools from Kaggle, GitHub, and other platforms.

Conclusion

The system effectively combines market research, AI use case generation, and resource collection into a streamlined application. Its modular and agent-based design ensures scalability and adaptability across industries.

System Requirements

Python 3.8+

Hugging Face API Key

DuckDuckGoSearchRun Integration

References

McKinsey AI Insights

Deloitte Digital Transformation Reports

Kaggle and GitHub resources for datasets

Future Work

Expand search capabilities with APIs like Serper for more refined insights.

Integrate deployment tools like Gradio for broader accessibility.

Enhance visualization of generated insights using charts and graphs.

Photos and links:

Streamlit : https://research-agents-eetunste6v8e8flzwcdak6.streamlit.app/