**Report: Market Research & Use Case Generation Agent for AI and Generative AI in Industries**

**1. Introduction**

**1.1 Project Overview**

This project aims to design and implement a **Multi-Agent architecture** to generate AI and **Generative AI (GenAI)** use cases for companies across various industries. The system is designed to conduct in-depth market research, understand the company's industry, and generate relevant AI use cases that improve operations and customer experience. It focuses on **AI/ML solutions** and how companies can leverage these technologies for **operational efficiency** and **customer satisfaction**.

**1.2 Objective**

The main goal is to build an architecture that:

1. Conducts market research for a given company or industry.
2. Identifies key strategic focus areas of the company.
3. Proposes relevant use cases where **GenAI, LLMs,** and **ML technologies** can be applied.
4. Collects resource assets (e.g., datasets, papers) related to the generated use cases.

**2. System Architecture**

**2.1 Architecture Overview**

The system comprises a **multi-agent framework** where each agent is responsible for a specific task. The agents interact with each other to generate a comprehensive list of AI use cases for the company/industry.

**2.2 Agents in the Architecture**

1. **Research Agent**:
   * Conducts research using a web browser tool to gather information on the industry, company, and market trends.
   * Segments the company by industry (e.g., Automotive, Finance, Healthcare).
2. **Use Case Generation Agent**:
   * Analyzes industry trends, standards, and competitors.
   * Proposes relevant use cases where **GenAI**, **LLMs**, and **ML technologies** can improve processes, customer satisfaction, and efficiency.
3. **Resource Collection Agent**:
   * Searches for relevant datasets and resources on platforms like **Kaggle**, **HuggingFace**, and **GitHub**.
   * Collects URLs and saves them in an organized format.

**2.3 Architecture Flowchart**

[Insert a diagram here showcasing the interaction between the agents. Use tools like draw.io to illustrate how each agent performs its task and shares the output.]

**3. Methodology**

**3.1 Industry Research**

The **Research Agent** performs web scraping using Selenium and other web scraping tools to gather the following information:

* **Industry Focus**: Understand the industry in which the company operates (e.g., Automotive, Finance).
* **Company Overview**: Identify the company’s key offerings, products, and services.
* **Strategic Focus Areas**: Research areas such as supply chain, operations, and customer experience.

**Example**: For Tesla, the research agent gathers information related to Tesla's focus on **electric vehicles (EVs)**, **sustainability**, and their **AI** initiatives in autonomous driving.

**3.2 Use Case Generation**

The **Use Case Generation Agent** analyzes the market trends and standards within the company’s sector and proposes AI/ML use cases, such as:

* AI-powered customer service chatbots.
* Predictive maintenance solutions for manufacturing.
* Intelligent automation for supply chain management.
* Personalized recommendations for retail businesses using AI/ML models.

**Example**: For the healthcare industry, the agent could suggest use cases like **AI-based diagnostic tools**, **automated patient record management**, or **predictive analytics for patient outcomes**.

**3.3 Resource Asset Collection**

The **Resource Collection Agent** performs a search for datasets and resources on platforms such as:

* **Kaggle**: Datasets related to healthcare, finance, or retail for AI models.
* **HuggingFace**: Pre-trained models for NLP tasks (e.g., chatbots, document summarization).
* **GitHub**: Open-source tools and libraries for implementing AI/ML solutions.

Links to relevant resources are collected in a text or markdown file for easy reference.

**4. Conclusion**

The **Multi-Agent architecture** successfully generated relevant AI use cases for a given industry. The proposed use cases align with industry trends and the company’s strategic needs, aiming to enhance **operational efficiency** and **customer satisfaction**. The agents also collected valuable datasets and resources to support the implementation of the proposed solutions.

**5. References**

1. **McKinsey & Company**: "AI in Retail: The Future of Customer Experience" [Link]
2. **Deloitte**: "AI Adoption in Healthcare: Trends and Opportunities" [Link]
3. **Kaggle**: Retail Datasets [Link]
4. **HuggingFace**: NLP Models for Customer Service [Link]

**6. Appendix**

**6.1 Source Code**

<https://github.com/VishnuVardhan280403/AI-Trends-in-Industry-A-Selenium-Web-Scraping-Approach>

**6.2 Demo Video**

<https://drive.google.com/file/d/1zqmfY7kOPYAa4bPp76IkD1uDn1ZaJ6yU/view?usp=sharing>

**6.3 Output\_Excel\_sheet**

[**https://docs.google.com/spreadsheets/d/1HbUlJ0zUjhhf5Xcsxj7ZnQQDNPN1Nwy3/edit?usp=sharing&ouid=113039736344459374998&rtpof=true&sd=true**](https://docs.google.com/spreadsheets/d/1HbUlJ0zUjhhf5Xcsxj7ZnQQDNPN1Nwy3/edit?usp=sharing&ouid=113039736344459374998&rtpof=true&sd=true)

**Source code:**

from selenium import webdriver

from selenium.webdriver.common.by import By

from selenium.webdriver.common.keys import Keys

from selenium.webdriver.chrome.service import Service

import time

from IPython.display import FileLink

def research\_industry\_with\_selenium(company\_name):

CHROMEDRIVER\_PATH = "C:\\Users\\T Vishnu vardhan\\OneDrive\\Desktop\\chromedriver\\chromedriver-win64\\chromedriver.exe"

service = Service(CHROMEDRIVER\_PATH)

options = webdriver.ChromeOptions()

options.add\_argument("--headless")

options.add\_argument("--no-sandbox")

options.add\_argument("--disable-dev-shm-usage")

driver = webdriver.Chrome(service=service, options=options)

try:

driver.get("https://www.google.com")

search\_box = driver.find\_element(By.NAME, "q")

search\_box.send\_keys(f"{company\_name} AI adoption and market trends")

search\_box.send\_keys(Keys.RETURN)

time.sleep(3)

search\_results = []

results = driver.find\_elements(By.CSS\_SELECTOR, "div.g")

for result in results[:10]:

try:

title = result.find\_element(By.TAG\_NAME, "h3").text

link = result.find\_element(By.TAG\_NAME, "a").get\_attribute("href")

search\_results.append({"title": title, "link": link})

except Exception as e:

continue #

return search\_results

finally:

driver.quit()

company\_name = "Tesla"

results = research\_industry\_with\_selenium(company\_name)

for idx, result in enumerate(results, start=1):

print(f"{idx}. {result['title']}\n {result['link']}")

output\_file = "tesla\_ai\_research.xlsx"

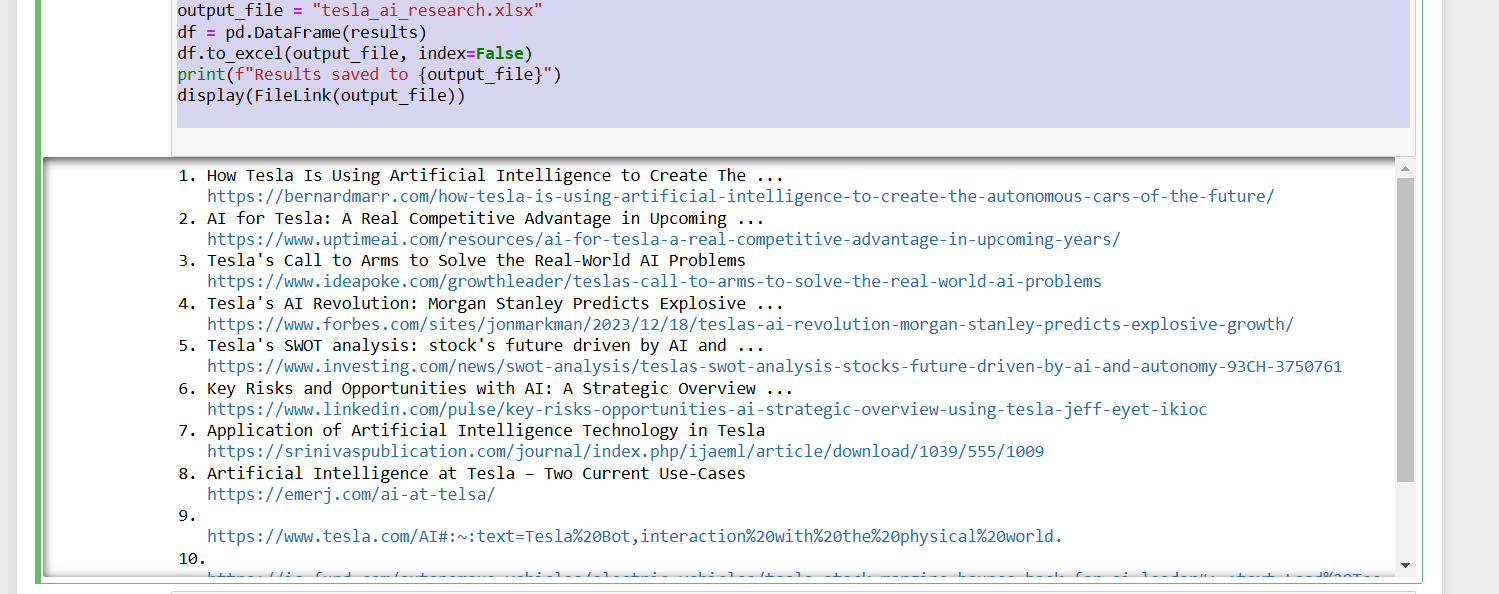
df = pd.DataFrame(results)

df.to\_excel(output\_file, index=False)

print(f"Results saved to {output\_file}")

display(FileLink(output\_file))

**Output:**

****