Web Research Agent Report

GitLink: https://github.com/ChandruKavi-Dev/Agentic_AI_Workshop/blob/e46c665293e9cd1d6b2c7dd031838bacca2c6eef/Day%204/Web%20Research%20Agent.zip

---

i. Brief Explanation: How LLM Was Used for Reasoning

The Large Language Model (LLM), specifically **Gemini Pro**, was used to perform the **reasoning (planning)** step of the ReAct (Reasoning + Acting) pattern. Given a user-defined topic, the LLM is prompted to:

- Understand the context of the topic.
- Generate **5–6 well-structured research questions** that explore various subtopics and dimensions (e.g., causes, impact, policies, trends).
- Ensure coverage across different aspects to guide the research agent during the web search phase.

This step simulates how a human might brainstorm areas to explore before starting their research. The LLM's natural language understanding allows the agent to formulate questions that are precise, informative, and diverse.

---

ii. Code and Flow of the Program

The program is structured in **four main stages**:

🔧 1. Setup

- Install dependencies: google-generativeai for Gemini and tavily-python for web search.
- Set API keys for Gemini and Tavily.

```
genai.configure(api_key=GEMINI_API_KEY)

model = genai.GenerativeModel("gemini-pro")
```

🧠 2. Reasoning Phase — generate_questions()

- Accepts the topic as input.
- Uses Gemini Pro to generate 5–6 relevant and diverse research questions.
- Returns a cleaned list of questions.

```
def generate_questions(topic):

    response = model.generate_content(prompt)

    ...
```

🌐 3. Acting Phase — search_answers()

- For each question, Tavily is used to perform live web searches.
- Extracts top 3 results (title + summary).
- Returns a compiled answer for each question.

```
def search_answers(question):
   results = tavily_client.search(...)
   ...
```

📄 4. Report Generation — compile_report()

- Combines the topic, questions, and answers into a structured markdown report.
- Sections: Title, Introduction, Questions + Answers, and Conclusion.
- Displayed using Colab's Markdown() viewer.

```
def compile_report(topic, qa_pairs):
   ...
   return markdown_text
```

✅ Final Execution Flow

At the end of the notebook:

- The user is prompted to enter a topic.
- The program generates questions, gathers answers, and displays the final report.

```
topic = input("🔍 Enter your research topic: ")
questions = generate_questions(topic)
...
display(Markdown(final_output))
```

This ReAct agent simulates a human-like research assistant using Gemini for planning and Tavily for tool-based acting, effectively combining LLM reasoning with real-time web information gathering.