Web Research Agent

LLM Reasoning & Program Flow Report

i. LLM Reasoning Explanation

This project uses the **ReAct (Reasoning + Acting)** paradigm to build an AI-powered web research agent. In the **Reasoning Phase**, we integrate **Google's Gemini Pro** LLM to generate 5–6 meaningful research questions based on a user-defined topic.

How the Reasoning Works:

- The user enters a topic (e.g., "Artificial Intelligence in Healthcare").
- The `generate_research_questions()` function sends a prompt to **Gemini** asking it to generate insightful and diverse questions on that topic.
- Gemini responds with multiple questions, typically structured as a list.
- These questions serve as a **research plan**, guiding what the agent will search for on the web next.

> This step demonstrates reasoning by encouraging the LLM to break the topic down into sub-questions across causes, effects, challenges, and solutions — effectively guiding the research like a human analyst.

ii.Code and Flow Explanation

The entire program follows a clean, modular structure:

1. **Folder Structure**

web_research_agent/

├── .env # API keys for Gemini and Tavily

├── app.py # Streamlit app entry

├── agent/

│ └── research_agent.py # Agent logic (plan, act, report)

├── utils/

│ ├── llm.py # Gemini LLM interaction

│ └── search.py # Tavily web search API

├── output/

│ └── research_report.md # Saved report

2. Component Breakdown

🔹 **utils/llm.py**

- Loads the Gemini API key.
- Sends the research topic to Gemini with a prompt like:

  "Generate 5–6 insightful research questions about the topic: 'Climate Change'."

- Parses the response into a clean Python list of questions.

🔹 **`utils/search.py`**

- Uses the **Tavily API** to search for each question online.
- Extracts titles and content snippets from the top 3 results per question.

🔹 **`agent/research_agent.py`**

- Combines the ReAct logic into three phases:
    - `plan()` → generate research questions.
    - `act()` → search and gather answers.
    - `compile_report()` → format the report with all data.

🔹 **`app.py`**

- Implements a clean
    - User inputs a topic.
    - Click “Enter”.
    - Final report is displayed in - `output/research_report.md`.

🌀 Flow Summary

User Inputs Topic → Gemini Generates Questions (Plan) →

Tavily Searches Web for Each Question (Act) →

Markdown Report Compiled (Report) →

Results Displayed in and Saved to Disk