Web Research Agent using ReAct Pattern

🔍 Objective

The goal of this assignment was to build a **Web Research Agent** that can intelligently plan, act, and generate a structured report on any user-defined topic using the **ReAct pattern (Reasoning + Acting)**. The agent utilizes:

- A **Large Language Model (LLM)** – Gemini – for planning.
- A **Web Search API** – Tavily – for real-time information retrieval.

🧠 Understanding ReAct Pattern

The **ReAct pattern** enables an agent to:

1. **Reason** about the task (generate a plan).
2. **Act** by using external tools (web search, APIs).
3. **Reflect** and compile a meaningful output.

This pattern allows the agent to operate in a goal-directed loop, rather than executing fixed instructions.

🏗️ System Architecture

✅ 1. Input

The agent accepts a single user-defined **topic** (e.g., "The Role of AI in Transforming Remote Healthcare Access in Rural India").

✅ 2. Reasoning Phase – Question Generation

The agent uses **Gemini (Google's LLM)** to generate 5–6 structured, diverse research questions related to the topic. This step ensures the research is organized and multifaceted.

Example Questions Generated:

- What are the current healthcare challenges in rural India?
- How is AI being used to improve medical diagnostics remotely?
- What are some AI-powered tools used in rural healthcare delivery?

✅ 3. Acting Phase – Web Search for Answers

Each generated question is passed to the **Tavily Web Search API**, which returns:

- Titles
- Content snippets of top relevant sources

To avoid API errors (like query length limits), the system automatically rephrases or shortens queries when needed.

The answers from multiple results are summarized and saved for each question.

✅ 4. Report Compilation

Finally, all the gathered data is compiled into a **structured research report**, which includes:

- Title
- Introduction
- Question-wise sections with summarized answers
- A conclusion

The report is formatted using markdown, making it easy to convert to PDF or HTML if needed.

🧩 Design Patterns Used

Pattern	Implementation

Planning	LLM (Gemini) generates relevant questions before starting research
Tool-Use	Agent chooses and uses Tavily API to find real-time web information
Loop Execution	Each question is processed individually in a loop to gather insights efficiently

---

🔧 Implementation Overview

Class-Based Design

The agent is implemented using a **Python class**, which handles the full flow:

- Accepts topic
- Generates questions
- Searches web
- Compiles report

Modular Functions

Key methods include:

- `generate_research_questions()`: Invokes Gemini to create research questions.
- `search_web_for_answers()`: Sends each question to Tavily and parses responses.
- `generate_report()`: Combines all content into a neatly formatted markdown report.

Error Handling

- Ensures Tavily query length is under 400 characters.
- Implements fallback strategies for long questions.
- Handles API errors gracefully.

💼 Execution Platform

- **Platform Used:** Google Colab
- **LLM Used:** Gemini via Google Generative AI SDK
- **Web Search:** Tavily API
- **Languages:** Python 3
- **Libraries:** `google.generativeai`, `tavily`, `dotenv`

✅ Conclusion

The Web Research Agent demonstrates an effective application of the ReAct pattern by combining planning (via LLM) and acting (via search tools). The modular and scalable design makes it easy to extend the agent for other research domains in the future.