Web Research Agent

Documentation

Submitted By: SANIGA BABU

Overview

This document explains the full architecture, tool integration, and inner workings of the **Web Research Agent** — a Python-based AI assistant that conducts autonomous web research.,

A powerful AI-driven assistant that searches the web, extracts key content, and summarizes it into a concise research report. Powered by Google Gemini, Serper.dev, and Streamlit.The Web Research Agent automates the entire process of online research: It analyzes a user's question, searches the web, extracts key information from pages, and generates a clean, summarized report — in minutes, with minimal user effort.

Agent Structure

Modular Components

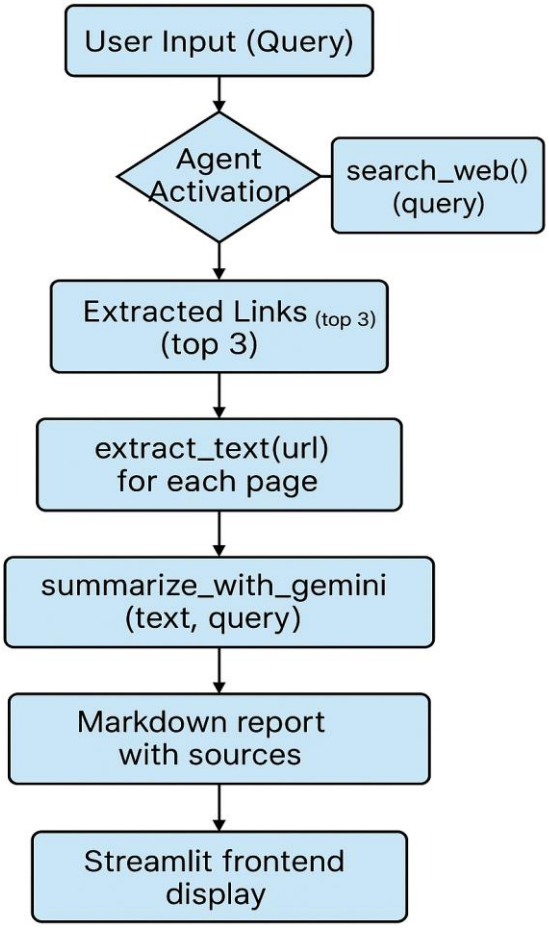
The agent is composed of the following key modules:

- main.py – Streamlit UI for user interaction
- agent.py – Orchestration logic and research flow
- search_tool.py – Web search API handler (Serper.dev)
- scraper.py – HTML content extractor
- summarizer.py – Google Gemini-based summarizer
- test_agent.py – Test script for CLI execution

Step-by-Step Flow

1. User enters a query via the Streamlit app.
2. run_agent() in agent.py is triggered.
3. The query is sent to the search_web() function.
4. Top 3 URLs are returned and iterated.
5. Each URL is passed to extract_text().
6. Extracted text is passed to summarize_with_gemini().
7. Summaries are combined into a markdown report.
8. Report is returned to the frontend.

Architecture Flow Chart



Prompt Design for AI Summarization

Objective

To generate accurate, relevant, and readable summaries from web content using Google Gemini.

Prompt Template

prompt = f"""
Summarize the following content in relation to the research question: '{query}'.
Focus on relevant facts, explanations, and useful details. Eliminate fluff.

Content:
{text}
"""

Prompt Goals

- Contextualize the content to the original query
- Focus on relevance, not just summarization
- Optimize for informative value over verbosity

Tool Integrations

1. search_tool.py – Web Search Tool

- **Input**: query (string)
- **Output**: List of (title, URL) tuples
- **External Service**: Serper.dev Google Search API
- **Use**: Finds relevant links to scrape based on user input

Error Handling:

- If response status is not 200, returns empty list
- Catches exceptions silently (to improve robustness)

2. scraper.py – Web Crawler / Content Extractor

- **Input**: A single url
- **Output**: Extracted plain text (top 5–10 paragraphs)
- **Libraries**: requests, BeautifulSoup
- **Use**: Converts raw HTML into readable content

Error Handling:

- Timeout of 5s
- Try/except for request and parsing errors
- Returns error message if failed

3. summarizer.py – Content Analyzer / Summarizer

- **Input**: Extracted text + original query + optional URL
- **Output**: Contextual summary (text)
- **Tool**: google.generativeai (Gemini 2.5)
- **Use**: Produces concise, relevant summaries of each page

Error Handling:

- Skips summarization if YouTube URL
- Can be expanded to handle long input token lengths

Decision Logic in agent.py

- Only summarizes pages that return valid content
- Skips pages with extraction failures
- Combines all summaries into a clean markdown report
- If no content is retrieved, informs the user explicitly

Error Handling Summary

Component	Error Strategy
Web Search	Empty list on API error, printed message
Web Scraping	Returns error message string, caught in agent
Summarization	Skips YouTube URLs; avoids long/invalid input
Main Agent	Catches and logs all unexpected exceptions

Summary Table – Tool I/O

Tool	Input	Output	Agent Use
search_web()	Query string	List of (title, URL) tuples	Finds relevant pages to investigate
extract_text()	Web page URL	Page text (string)	Extracts readable content
summarize_with_gemini()	Page text + query	Summary (string)	Produces contextual research summaries

Conclusion

This Web Research Agent provides a modular, extensible foundation for automated internet research. With improvements, it can scale into an academic, enterprise, or journalistic-grade assistant.