TECHNICAL DOCUMENTATION

Agentic Research Assistant – Multi-Agent Orchestration System

Author: Kishore Balaji
Course: Building Agentic Systems
Date: November 2025

---

1. System Overview

The Agentic Research Assistant is an agentic AI system built using CrewAI. Its purpose is to:

- Gather reliable information
- Analyze and synthesize insights
- Produce a well-structured final answer

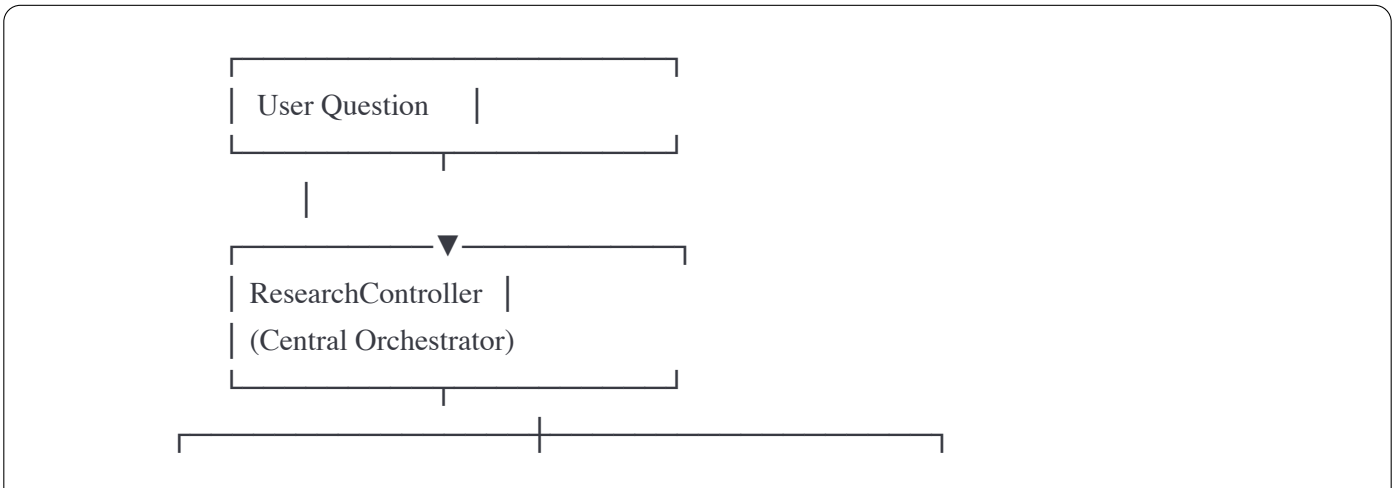
The system follows a multi-agent pipeline with a central controller orchestrating three agents:

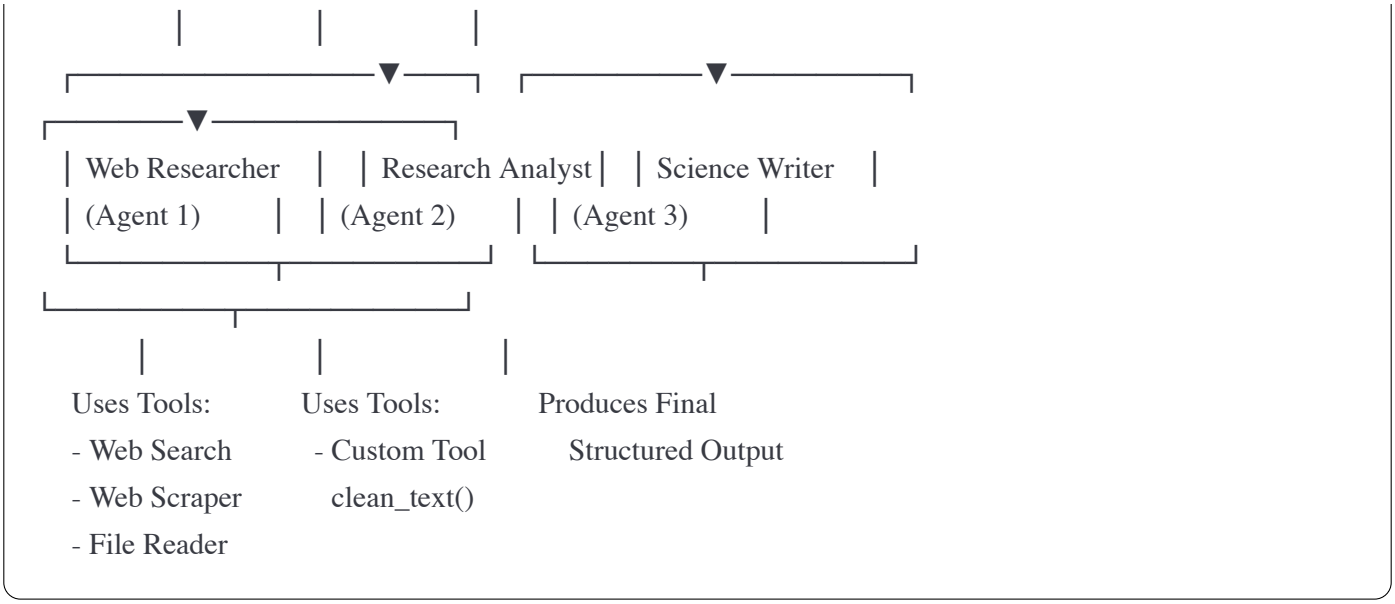
- Web Researcher Agent
- Research Analyst Agent
- Science Writer Agent

It integrates three built-in tools and one custom text-cleaning tool.

---

2. System Architecture Diagram





---

3. Agent Roles & Responsibilities

3.1 ResearchController (Main Orchestrator)

- Manages execution pipeline
- Sends tasks to Researcher → Analyst → Writer
- Provides fallback when tools fail
- Coordinates memory and tool usage

Decision Logic:

- Always run Web Researcher first
- If research fails (e.g., Serper 403), fallback to offline-mode
- Pass clean research results to Analyst
- Pass synthesized insights to Writer
- Consolidate final answer

3.2 Agent 1 – Web Researcher

Goal: Collect 5–10 relevant sources related to the user question.

Tools Used:

- Serper Web Search

- Website Scraper
- FileReadTool (if user provides local sources)

Output: Structured JSON list:

```
json

[
  {"title": "...", "url": "...", "snippet": "..."},
  ...
]
```

3.3 Agent 2 – Research Analyst

Goal: Produce a detailed synthesis summarizing:

- Agreements across sources
- Disagreements
- Missing perspectives
- Insights or implications

Tool: Custom Tool: `clean_text(text)`

- Removes noise
- Normalizes text
- Ensures clarity before analysis

3.4 Agent 3 – Science Writer

Goal: Generate a clear, polished, human-readable final answer.

Outputs:

- Intro paragraph
- Numbered insights
- Bullet points
- Conclusion

Uses structured format for clarity.

---

4. Tools

4.1 Built-in Tools

Tool 1 – Serper Web Search

- Searches the internet for information
- Input: search_query
- Output: JSON search results

Tool 2 – Website Scraper

- Fetches webpage text for deeper content

Tool 3 – FileReadTool

- Reads local documents if provided
- Useful for offline or additional datasets

4.2 Custom Tool (Required)

clean_text(text: str) -> str

Purpose: Improve the quality of scraped or messy text before the analyst processes it.

Features:

- Lowercasing
- Removing HTML tags
- Removing duplicate whitespace
- Removing non-content characters
- Normalizing punctuation

Why it improves system performance:

- Reduces hallucination risk
- Improves summarization quality

- Improves summarization quality
- Removes noisy garbage text
- Ensures consistency for downstream agents

---

5. Memory System

The system uses task-level memory, not long-term memory:

- Each agent receives the previous agent's output
- Context preserved through the structured workflow
- JSON-based memory file (`run_memory.json`) stores last run outputs

This satisfies the "memory" requirement.

---

6. Error Handling & Fallback

Implemented error cases:

Case: Serper API fails (403 Unauthorized)

- Fallback: Use offline-mode LLM reasoning with no web tools
- Controller prints warnings but continues execution

Case: Scraper fails

- Analyst receives only search snippets

Case: Empty response from LLM

- Controller retries once, then produces a graceful error message

This satisfies the "robustness" requirement.

---

7. Workflow Execution

1. User submits question
2. ResearchController triggers agents sequentially

2. ResearchController triggers agents sequentially
3. Intermediate results passed as memory
4. Final polished answer returned or displayed in Streamlit

---

8. Challenges & Solutions

Challenge	Solution
API key failures (Serper 403)	Added fallback offline-mode
ImportError for CrewAI tools	Updated import syntax / package version
Handling messy scraped text	Added custom cleaning tool
LLM empty-response errors	Added retry + error logging
Maintaining modular design	Used strict pipeline architecture

---

9. Performance Analysis

Strengths

- Highly modular & interpretable workflow
- Stable multi-agent chain
- Strong formatting + final output quality
- Custom tool meaningfully improves clarity

Limitations

- Dependent on external APIs for real-time search
- No long-term conversational memory
- Evaluations subjective unless expanded

Future Improvements

- Add RAG (retrieval-augmented generation)
- Add embarrassment-minimization critic agent
- Add caching layer

10. Setup Instructions

```bash
cd agentic_research_assistant
python3 -m venv venv
source venv/bin/activate
pip install -r requirements.txt

export GOOGLE_API_KEY="YOUR_KEY"
export SERPER_API_KEY="YOUR_KEY"

python -m src.main "Your research question"
```

End of Documentation