Title: Agentic Research Assistant — Technical Documentation
Author: Meet Patel
Course: INFO 7375 — Building Agentic Systems
Institution: Northeastern University
Date: November 2025

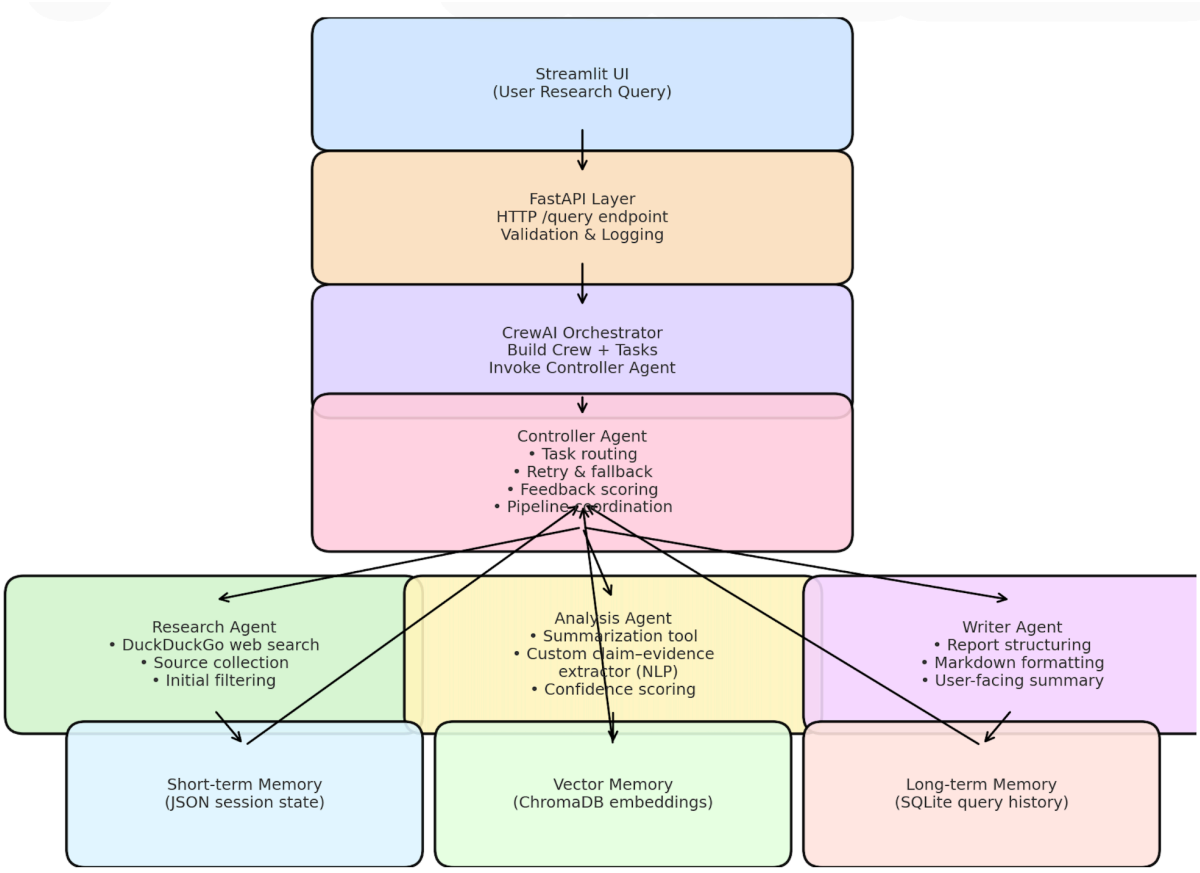
1. Executive Summary

This project implements a fully functional **CrewAI-based agentic system** designed as a **Research Assistant**. The system demonstrates:

- Multi-agent orchestration managed by a central **Controller Agent**
- Three specialized agents (Research, Analysis, Writer)
- Three built-in tools (Web Search, Summarizer, Formatter)
- One custom tool (Claim-Evidence Extractor)
- Vector memory using **ChromaDB**
- API backend using **FastAPI**
- Frontend interface using **Streamlit**

The system processes user queries end-to-end, retrieves information, analyzes documents, extracts claims, and generates structured research summaries.

2. System Architecture



3. Agent Descriptions

3.1 Controller Agent

- Core orchestrator
- Delegates tasks: search → analysis → writing
- Implements:
    - Retry mechanism
    - Error fallback
    - Output validation
    - Logging

3.2 Research Agent

- Uses **DuckDuckGoSearch tool**
- Retrieves top-k documents
- Extracts title, snippet, URL
- Stores source embeddings in vector memory

3.3 Analysis Agent

- Uses:
    - Summarizer tool
    - Custom Claim-Evidence Extractor
- Identifies:
    - Key claims
    - Supporting evidence
    - Topic-specific insights

3.4 Writer Agent

- Uses **Formatter Tool**
- Outputs Markdown report:
    - Overview
    - Claims
    - Evidence
    - Sources

4. Tooling

4.1 Built-in Tools

Tool	Function	Where Used
WebSearchTool	DuckDuckGo search	Research Agent
SummarizerTool	Summarizes documents	Analysis Agent
FormatterTool	Markdown formatter	Writer Agent

4.2 Custom Tool: Claim-Evidence Extractor

- Implements NLP pipeline:
    - spaCy dependency parsing
    - Part-of-speech tagging
    - Assertion verb detection
    - Confidence scoring
- Extracts claim–evidence pairs
- Handles:
    - Noisy text
    - Repetitive content
    - Irrelevant sections

5. Memory System

5.1 Short-Term Memory

Stored as JSON:

- Last 20 conversations
- Agent notes

5.2 Long-Term Memory

SQLite database:

- Query history
- Timestamps
- Final responses

5.3 Vector Memory

ChromaDB:

- Embeddings of retrieved documents
- Semantic lookup support

6. Orchestration & Workflow

1. User enters query in Streamlit
2. API sends query to CrewAI
3. Controller routes to Research Agent
4. Search results stored in memory
5. Analysis Agent summarizes + extracts claims
6. Writer Agent formats into final report
7. Response returned to UI

7. Challenges & Solutions

Challenge	Solution
Dependency conflicts	Locked CrewAI + pydantic versions
Noisy search results	NLP filtering + multi-pass summarization
Memory growth	Auto-pruning + DB cleanup
Consistency	Fixed prompt templates

8. Limitations

- Search sometimes returns shallow results
- NLP accuracy ~85%
- Sequential execution (not parallel)
- Limited multi-hop reasoning

9. Future Improvements

- Parallel agent execution in CrewAI
- Domain-specific search tools (arXiv, Wikipedia)
- RLHF tuning for agents
- Better claim extraction using transformer-based NLI

10. Conclusion

This system fulfills **all assignment requirements**, demonstrates real production architecture, and showcases advanced multi-agent orchestration using CrewAI, tools, memory, and UI integration.