Jade Sanchez

2376 ITAI

5/6/2025

Source Code Repository

research_agent_main.py

```
import os import requests from typing import List, Dict
```

Simulated Memory System

```
class AgentMemory: def init(self): self.memory = {}
```

```
def store(self, key: str, value: str):
    self.memory[key] = value

def retrieve(self, key: str) -> str:
    return self.memory.get(key, "")
```

Tool 1: Web Search (Mock)

```
def web_search(query: str) -> List[str]: print(f"Searching the web for: {query}") return [ f"Result 1 for {query}", f"Result 2 for {query}", f"Result 3 for {query}" ]
```

Tool 2: Summarizer (Mock)

```
def summarize(texts: List[str]) -> str: return " ".join([f"Summary of: {text}" for text in texts])
```

Reasoning & Planning

```
class ResearchAgent: def init(self): self.memory = AgentMemory()
```

```
def process_input(self, topic: str):
    if not topic:
        raise ValueError("Invalid topic input.")
    return topic.strip().lower()

def plan_actions(self, topic: str) -> Dict:
    return {"search_query": f"{topic} latest research"}

def execute_plan(self, plan: Dict) -> str:
    results = web_search(plan['search_query'])
    summary = summarize(results)
    return summary

def get_feedback(self, summary: str) -> int:
    print("User feedback requested: rate summary 1 (poor) to 5 (excellent)")
    return 4  # mock positive feedback

def improve_policy(self, feedback: int):
    if feedback < 3:
        print("Agent learns to search deeper sources next time.")

def generate_output(self, summary: str) -> str:
    return f"Organized Report:\n\n{summary}\n"

def research_topic(self, topic: str) -> str:
    topic = self.process_input(topic)
    plan = self.plan_actions(topic)
    summary = self.execute_plan(plan)
    self.memory.store(topic, summary)
    feedback = self.get_feedback(summary)
    self.improve_policy(feedback)
    return self.generate_output(summary)
```

Safety Checks

```
def validate_topic(topic: str) -> bool: blocked_terms = ["violence", "malware", "hacking"] return
all(term not in topic for term in blocked_terms)

if name == "main": agent = ResearchAgent()

user_topic = input("Enter your research topic: ")

if validate_topic(user_topic):
    report = agent.research_topic(user_topic)
    print(report)
else:
    print("Topic not allowed for safety reasons.")
```