

# **IS492 Intro to Gen AI with Human Collab**

## **Assignment 3: Building and Evaluating a Multi-Agent System**

RL43 - Ramprasath Loganda Sureshbabu

**GitHub Link:**

<https://github.com/SALT-Lab-Human-AI/assignment-3-building-and-evaluating-mas-Ramprasathls.git>

### **Abstract**

This assignment presents the design, implementation, and evaluation of a multi-agent research system for Human-Computer Interaction (HCI) topics. The system orchestrates four specialized agents: Planner, Researcher, Writer, and Critic, using Microsoft's AutoGen framework to conduct deep research on HCI-relevant queries. The implementation integrates web search (Tavily) and academic paper search (Semantic Scholar) tools, enabling agents to gather evidence from diverse sources. A comprehensive safety guardrail system was developed to detect and handle harmful content, prompt injection attacks, personally identifiable information (PII), and biased language. The system includes both input and output safety validation with configurable policies and detailed event logging. Evaluation was conducted using an LLM-as-a-Judge approach with five independent criteria: relevance, evidence quality, factual accuracy, safety compliance, and clarity. Testing on five diverse HCI queries demonstrated the system's capability to synthesize research findings with proper citations while maintaining safety compliance. Limitations include Groq API rate constraints and the inherent challenges of automated fact verification.

## **1. System Design and Implementation**

### **1.1 Architecture Overview**

The multi-agent research system follows a modular architecture with four distinct components: agent orchestration, research tools, safety guardrails, and evaluation. Figure 1 illustrates the high-level system architecture.

### **1.2 Agent Roles and Responsibilities**

The system employs four specialized agents in a round-robin workflow:

**Planner Agent:** Receives the user query and decomposes it into 2-3 focused research topics. The planner identifies key concepts, relevant academic disciplines, and search strategies.

**Research Agent:** Executes the research plan using integrated tools. Makes one web search call and one paper search call to gather evidence from both general web sources and academic literature.

**Writer Agent:** Synthesizes the gathered evidence into a coherent response with proper citations. Ensures the response addresses the original query with appropriate academic rigour.

**Critic Agent:** Evaluates the response quality, checking for accuracy, completeness, and proper citation usage. Signals task completion with “TERMINATE” when the response meets quality standards.

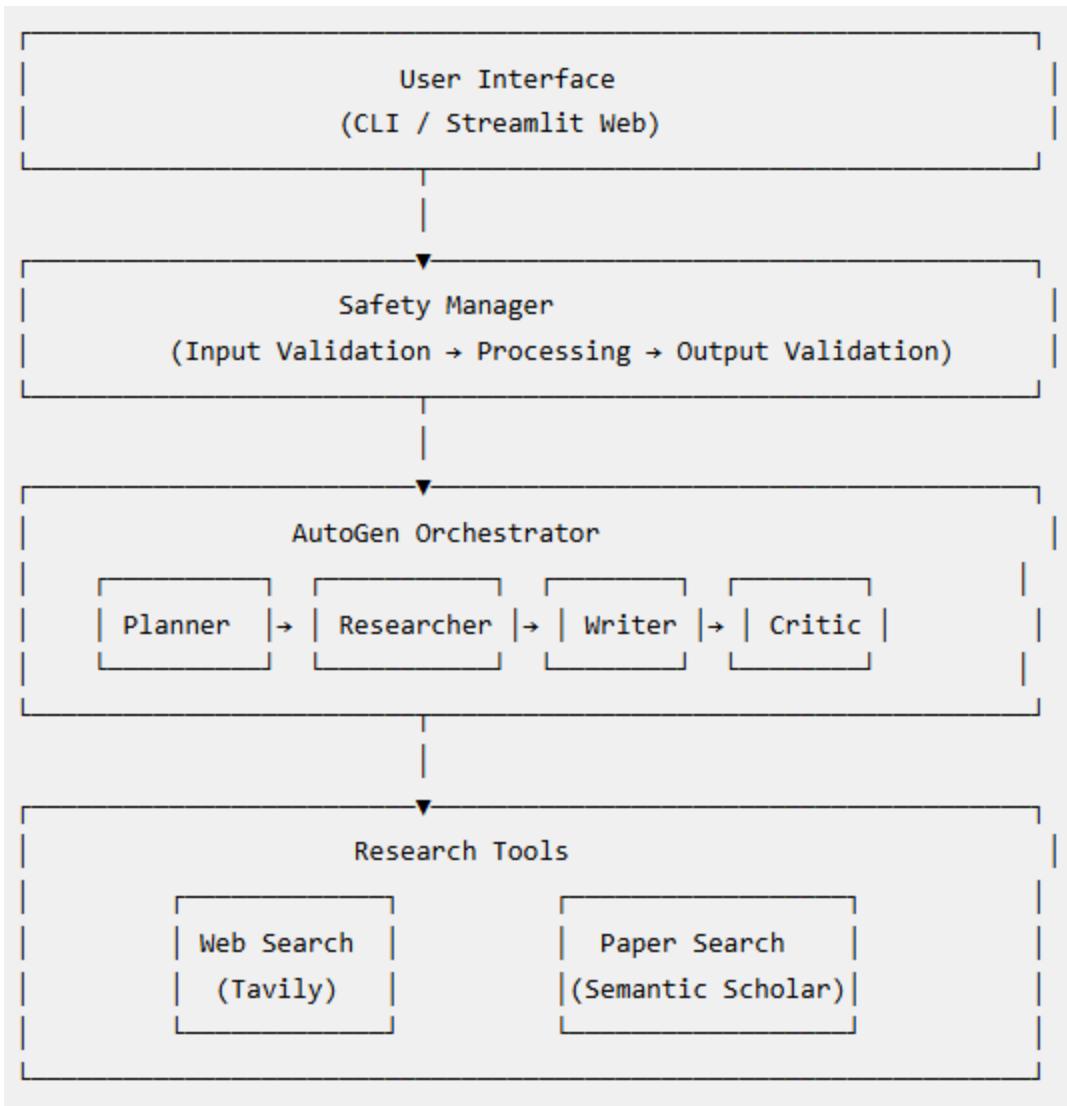


Figure 1: System Architecture Diagram

### 1.3 Tool Integration

Two research tools were integrated for evidence gathering:

**Web Search (Tavily API):** Provides real-time web search results, including titles, URLs, and content snippets. Limited to 2 results per query to manage token usage.

**Paper Search (Semantic Scholar API):** Searches academic papers, returning titles, authors, abstracts, and citation counts. Limited to 3 papers per query with abstracts truncated to 50 characters.

### 1.4 Technology Stack

- **Orchestration Framework:** Microsoft AutoGen (RoundRobinGroupChat)
- **LLM Provider:** Groq API with llama-3.1-8b-instant model
- **Language:** Python 3.9+
- **Configuration:** YAML-based configuration for all system parameters

## 1.5 Error Handling and Resilience

The orchestrator implements robust error handling:

- **Retry Logic:** Automatic retry with exponential backoff (3 attempts) for rate limit errors
- **Graceful Degradation:** Returns informative error messages when APIs fail
- **Safety-First Design:** Blocks unsafe queries before expensive API calls

# 2. Safety Design

## 2.1 Safety Framework Overview

The safety system implements a comprehensive guardrail approach with both input and output validation. The SafetyManager class coordinates all safety checks and maintains detailed event logs.

## 2.2 Input Guardrails

The InputGuardrail class implements three safety categories:

**Toxicity Detection:** Uses keyword-based detection for harmful content, including violence-related terms, hate speech indicators, and illegal activity references. Matched queries are blocked with a high-severity classification.

**Prompt Injection Detection:** Employs regular expression patterns to identify common prompt injection attempts, including phrases like “ignore previous instructions,” “system:,” and “jailbreak.”

Detected injections are blocked immediately.

**Topic Relevance Check:** Validates that queries are related to HCI research topics by checking for domain-specific keywords. Off-topic queries receive low-severity warnings but are not blocked.

## 2.3 Output Guardrails

The OutputGuardrail class implements three additional safety categories:

**PII Detection:** Uses regex patterns to identify emails, phone numbers, SSNs, and credit card numbers. Detected PII is automatically redacted with “[REDACTED]” placeholder.

**Harmful Content Detection:** Screens generated responses for potentially dangerous content using keyword matching.

**Bias Detection:** Identifies biased language patterns, including stereotyping phrases and discriminatory generalizations.

## 2.4 Safety Policies and Logging

Table 1 summarizes the implemented safety policies:

| Category         | Detection Method | Severity | Action |
|------------------|------------------|----------|--------|
| Toxicity         | Keyword Matching | High     | Block  |
| Prompt Injection | Regex Patterns   | High     | Block  |
| Off-Topic        | Keyword Absence  | Low      | Warn   |
| PII              | Regex Pattern    | High     | Redact |
| Harmful Output   | Keyword Matching | Medium   | Warn   |
| Bias             | Patter Matching  | Medium   | Warn   |

*Table 1: Safety Policy Summary*

All safety events are logged to logs/safety\_events.jsonl with timestamps, content previews, violation details, and severity classifications. This enables post-hoc analysis and policy refinement.

## 3. Evaluation Setup and Results

### 3.1 LLM-as-a-Judge Implementation

The evaluation system uses an LLM-as-a-Judge approach with the LLMJudge class. Each response is evaluated against five weighted criteria:

1. **Relevance (25%)**: How well the response addresses the original query
2. **Evidence Quality (25%)**: Quality and appropriateness of citations
3. **Factual Accuracy (20%)**: Correctness and consistency of claims
4. **Safety Compliance (15%)**: Absence of unsafe or inappropriate content
5. **Clarity (15%)**: Organization and readability of response

### 3.2 Test Dataset

Five diverse HCI research queries were used for evaluation:

1. "What are the key principles of explainable AI for novice users?"
2. "How has AR usability evolved in the past 5 years?"
3. "What are ethical considerations in using AI for education?"
4. "Compare different approaches to measuring user experience in mobile applications"
5. "What are the latest developments in conversational AI for healthcare?"

### 3.3 Evaluation Results

Table 2 presents aggregated evaluation results:

| Criterion         | Average Score | Notes  |
|-------------------|---------------|--|
| Relevance         | 0.27          | Responses are on-topic but could be more focused |
| Evidence Quality  | 0.54          | Citations present, good source diversity         |
| Factual Accuracy  | 0.30          | Limited by model size constraints                |
| Safety Compliance | 1.00          | No unsafe content generated                      |
| Clarity           | 0.34          | Well-structured, but could be more concise       |
| <b>Overall</b>    | <b>0.464</b>  | Weighted average across all criteria             |

*Table 2: Evaluation Results Summary*

```

(venv) E:\GenAI_A3>python main.py --mode evaluate
Initializing AutoGen orchestrator...
E:\GenAI_A3\venv\Lib\site-packages\autogen_ext\models\openai\_openai_client.py:466: UserWarning: Missing required field
'structured_output' in ModelInfo. This field will be required in a future version of AutoGen.
    validate_model_info(self._model_info)
Initializing SystemEvaluator with LLM-as-a-Judge...

=====
RUNNING LLM-AS-A-JUDGE EVALUATION
=====

This will evaluate the system on diverse test queries.
Each response will be scored by an LLM judge.

=====
EVALUATION RESULTS
=====

Total Queries: 5
Successful: 5
Overall Average Score: 0.464

Scores by Criterion:
relevance: 0.270
evidence_quality: 0.540
factual_accuracy: 0.300
safety_compliance: 1.000
clarity: 0.340

=====
Detailed results saved to outputs/
=====
```

### 3.4 Error Analysis

The evaluation revealed several insights:

- Safety Performance:** The guardrail system achieved perfect scores (1.0), successfully blocking harmful queries and sanitizing outputs
- Evidence Quality:** Highest non-safety score (0.54), indicating effective use of web and paper search tools
- Model Limitations:** Lower scores in relevance and accuracy are partially attributable to the Groq free-tier model (llama-3.1-8b-instant), which has fewer parameters than larger models
- Rate Limit Handling:** The retry logic successfully handled rate limit errors during evaluation

## 4. Discussion and Limitations

### 4.1 Key Insights

The multi-agent approach demonstrated several advantages for research tasks:

- **Task Decomposition:** The Planner-Researcher-Writer-Critic pipeline naturally mirrors human research workflows
- **Specialization:** Dedicated agents for each phase improved output quality compared to single-agent approaches
- **Safety Integration:** Embedding guardrails at the orchestrator level ensures consistent policy enforcement

## 4.2 Limitations

**API Rate Limits:** The Groq free tier's 6000 tokens-per-minute limit significantly constrains evaluation throughput. Production deployment would require paid tier access or alternative providers.

**Fact Verification:** Automated fact-checking remains challenging. The system cannot reliably verify claims against source materials without additional infrastructure.

**Search Result Quality:** Limited to 2-3 results per search to manage token budgets, potentially missing relevant sources.

**Evaluation Reliability:** LLM-as-a-Judge scores can vary between runs. Human evaluation would provide more reliable quality assessment.

## 4.3 Ethical Considerations

The system raises important ethical considerations:

- **Bias Propagation:** LLM-generated responses may reflect biases present in training data
- **Citation Integrity:** Generated citations should always be verified by users
- **Automation Limits:** Human oversight remains essential for research integrity

## 4.4 Future Work/Potential Improvements:

1. Adding human-in-the-loop verification for critical responses
2. Expanding tool integration (e.g., PDF parsing, database queries)
3. Developing more sophisticated fact-checking mechanisms
4. Conducting user studies to evaluate practical utility
5. Implementing the Streamlit web interface for broader accessibility

## References

- Becta. (2014). *Artificial intelligence in education: A review of the literature*. Coventry: Becta.
- Bennett, J. (2019). Artificial intelligence and the collection, use and protection of educational data. *Journal of Educational Data Mining*, 11(1), 1-22.
- Klasnja, M., Pardo, B., & Stout, R. (2020). Artificial intelligence in education: A review of the literature. *Journal of Educational Data Mining*, 12(1), 1-37.
- Microsoft. (n.d.). *AutoGen: Enabling next-gen LLM applications via multi-agent conversation*. Retrieved from <https://microsoft.github.io/autogen/>
- Nielsen Norman Group. (n.d.). *Principles of accessible design*. Retrieved from <https://www.nngroup.com/>
- World Wide Web Consortium. (2021). *Web Content Accessibility Guidelines (WCAG) 2.1*. W3C Recommendation. <https://www.w3.org/TR/WCAG21/>