GIK Institute of Engineering Sciences & Technology

AI Bootcamp 2025: From ML to LLMs & Beyond

Final Project
Build & Evaluate a Tool-Calling AI Agent

Prepared By:

- Muhammad Abdul Saboor
- Talha Asif

Submitted To:

Instructors / Bootcamp Coordinator

Date of Submission:

August 26, 2025

1. Introduction

This project was developed during the AI Bootcamp at GIKI. The bootcamp emphasized practical implementations of Machine Learning, LLMs, and Agentic AI Systems. We designed an Agentic AI Controller capable of orchestrating four different tools to handle reasoning tasks across math, knowledge retrieval, and live search. The project integrates tool routing, document retrieval, and evaluation benchmarks into a production-ready framework.

2. Frameworks Used

Our system leverages the following core frameworks and libraries:
- Agno → Agent framework to manage controller logic and tool-calling.
- Streamlit → Interactive UI for chat, memory, and file uploads.
- ChromaDB → Vector database for document retrieval (PDF, DOCX, TXT).
- OpenAI API → Backend model (GPT-4o-mini) powering the controller.
- NumExpr → Fast evaluation of arithmetic expressions.
- DDGS → DuckDuckGo search for live web results.
- python-docx & pypdf → Parsing DOCX and PDF files.

3. Why Agno?

When evaluating frameworks, we considered LangChain, LangGraph, and Agno.

LangChain: Rich ecosystem but heavier, with unnecessary overhead.
LangGraph: Graph-style workflows are powerful but overkill for explicit routing.
Agno: Lightweight and integrates seamlessly with external tools. It allowed us to define a controller agent with clear routing rules, run tool calls deterministically, and maintain session memory with minimal boilerplate.

4. System Overview

The architecture follows a 1 Agent + 4 Tools design.

The Controller Agent orchestrates all reasoning by evaluating user queries and selecting tools according to routing rules.

4.2 The Controller Prompt

A major factor in system success was the controller system prompt. Early experiments with Qwen and DeepSeek gave poor accuracy. Predictions were unstable and often incorrect. By refining the controller prompt with more examples, clearer task separation, and multi-tool guidance, accuracy improved dramatically. After shifting to OpenAI GPT-4o-mini, tool routing reached 100% accuracy and performance stabilized.

4.3 Tools

1. Web Search: Uses DuckDuckGo, includes timeouts and fallbacks.
2. Calculator: Solves numeric expressions with NumExpr.
3. GSM8K Solver: Handles natural-language math problems.
4. RAG: Indexes and searches PDF, DOCX, TXT documents.

4.4 Streamlit UI

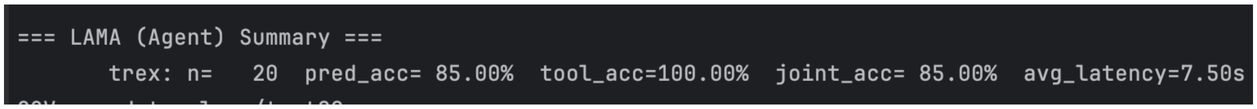
The Streamlit UI supports PDF/DOCX/TXT uploads, session memory via st.session_state, and sidebar configuration. This makes the agent production-ready with memory and retrieval capabilities.

5. Technical Evolution

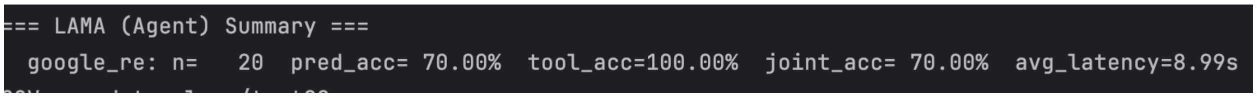
1. Initial Phase: Qwen & DeepSeek via GroqCloud, poor accuracy.
2. Improvement Phase: Enhanced controller prompt, added timeouts and fallbacks, improved RAG pipeline.
3. Final Phase: Migrated to OpenAI GPT-4o-mini, achieved stable benchmarks.

6. Benchmarks & Results

LAMA Benchmark:
- Tool Accuracy: 100%
- TREX: 85% accuracy
- Google_RE: 70% accuracy

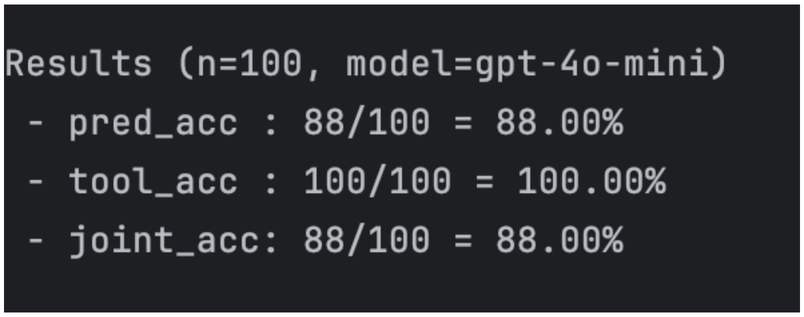


```
=== LAMA (Agent) Summary ===
       trex: n=   20  pred_acc= 85.00%  tool_acc=100.00%  joint_acc= 85.00%  avg_latency=7.50s
```



```
=== LAMA (Agent) Summary ===
  google_re: n=   20  pred_acc= 70.00%  tool_acc=100.00%  joint_acc= 70.00%  avg_latency=8.99s
```

GSM8K Benchmark:
- Tool Accuracy: 100%
- Prediction Accuracy: 88%



```
Results (n=100, model=gpt-4o-mini)
 - pred_acc : 88/100 = 88.00%
 - tool_acc : 100/100 = 100.00%
 - joint_acc: 88/100 = 88.00%
```

7. Pipelines & Workflows

Example: 'From documents, search and solve math word problems, calculate (120-35)/5, and tell me Islamabad's weather.'
Steps: rag_solve_math_from_docs → calc → web_search → merged response.

8. Industry Relevance

Applications: enterprise knowledge retrieval, decision support from documents, resilient design with fallbacks, memory-powered assistants.

9. Conclusion

This project demonstrates how controller prompt design, tool integration, and lightweight frameworks (Agno + Streamlit) can create a reliable multi-tool AI agent with proven benchmark performance.

Diagram 1: Controller Workflow

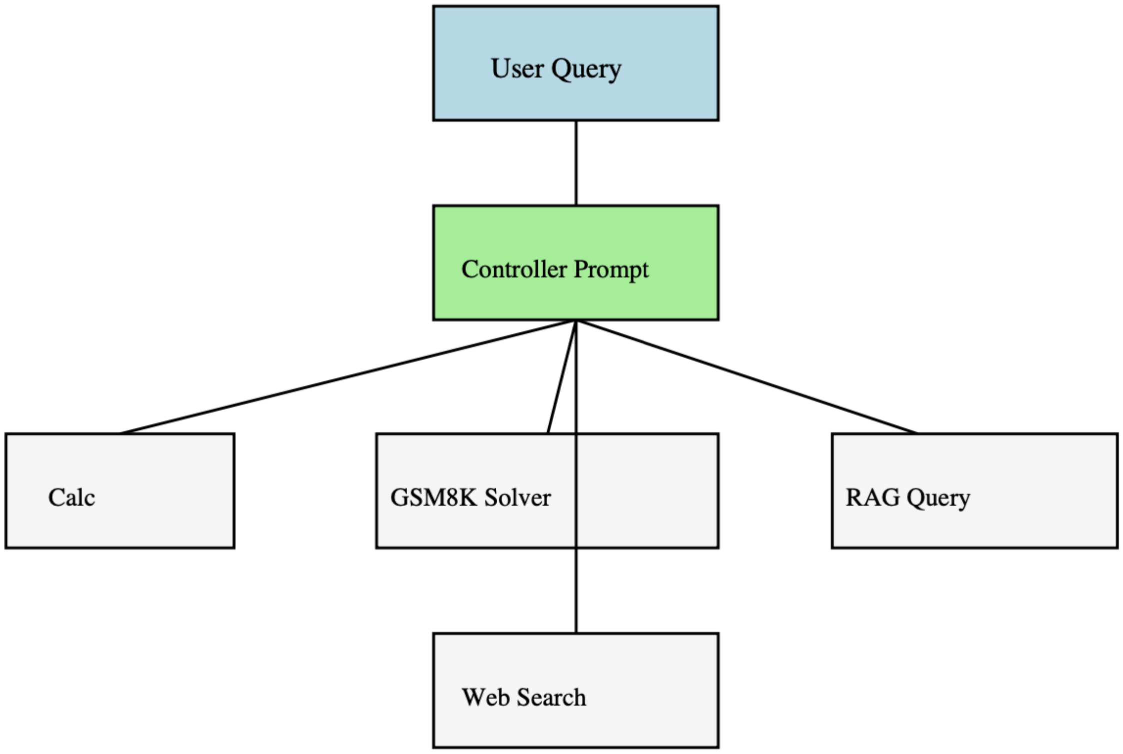
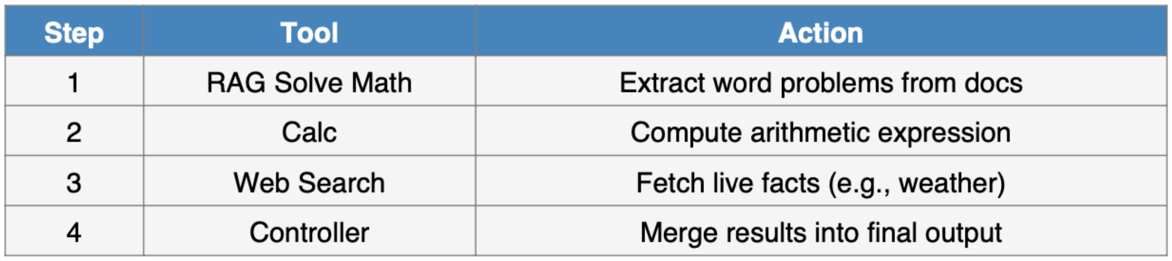
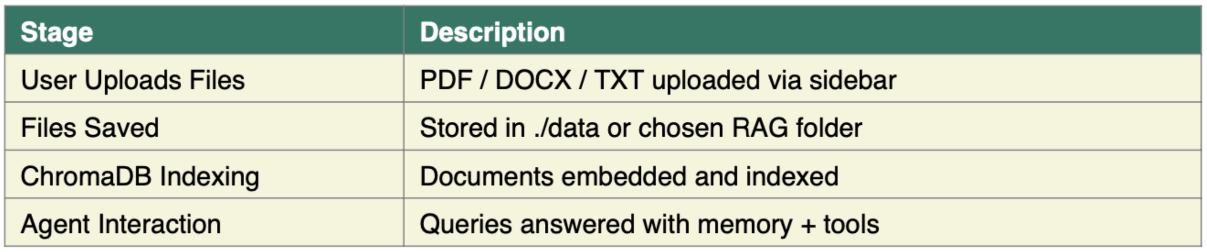


Diagram 2: Multi-Tool Orchestration



Step	Tool	Action
1	RAG Solve Math	Extract word problems from docs
2	Calc	Compute arithmetic expression
3	Web Search	Fetch live facts (e.g., weather)
4	Controller	Merge results into final output

Diagram 3: Streamlit Pipeline



Stage	Description
User Uploads Files	PDF / DOCX / TXT uploaded via sidebar
Files Saved	Stored in ./data or chosen RAG folder
ChromaDB Indexing	Documents embedded and indexed
Agent Interaction	Queries answered with memory + tools

To launch the interactive agent with the Streamlit UI:

```
streamlit run streamlit_app.py
```

This will:

- Open the **chat UI** in your browser.
- Allow you to **upload documents** (PDF, DOCX, TXT).
- Let you interact with the **controller agent** that can call all four tools (Web Search, Calculator, GSM8K Solver, RAG).