Project P5: Multi-Agent Knowledge Navigator

New Attempt

- Due Dec 16, 2025 by 11:59pm
- Points 100
- Submitting a media recording or a file upload

Overview

This is part 5 of our semester-long project: Intelligent Human Assistant. P5 brings together everything from P1-P4 to create a complete digital knowledge navigator. You now have natural language terminal (P1), powerful language models (P2), RAG-based knowledge management (P3), and computer control (P4). P5 integrates these capabilities into a unified system with a conversational interface.

Unlike previous assignments where you worked with individual components, P5 composes them into a cohesive system. Your terminal becomes an MCP tool, your fine-tuned models are replaced with powerful cloud models, your RAG system uses Open WebUI's built-in capabilities, and your computer control uses coordinate-based automation.

This composition approach is how modern AI systems integrate multiple capabilities. You will build a working version.

Learning Objectives

By completing this assignment, you will:

- Integrate multiple specialized systems into a unified interface
- Expose terminal functionality as MCP tools
- Configure and use powerful cloud-based models
- Leverage built-in RAG with advanced document parsing
- Implement coordinate-based UI automation
- Build a production-ready knowledge navigator

Core Components

You will implement and integrate five main components:

1. P1 as MCP Tool (Terminal)

Implementation:

- Wrap P1 terminal as MCP server (make sure to use a different port, e.g., 3003)
- Define tool schema for command execution

- Hint: We discussed how to run MCP servers and add them in OpenWebUI during in-class activity. Briefly
  - Use different ports for different MCP servers (e.g., port 3000 for P1, 3004 for P4)
  - Use different url:port/mcp to add different server to OpenWebUI

Operations:

execute_command(cmd: str) -> output

2. Cloud Models (Replacing P2)

Options:

- ai (GPT-4, Claude, etc.)
- gpt-oss:20b-cloud

Setup: Configure API credentials in Open WebUI settings

3. Built-in RAG (Replacing P3)

Configuration:

- Enable RAG in Open WebUI settings
- Configure Mistral OCR parsing
- Upload documents through Open WebUI
- Hints: We discussed how to do it during in-class activity. Briefly,
  - In Open-WebUI, go to Workspace > Knowledge.
  - Click the + icon to create a new knowledge base (KB).
  - Enter a name and description when prompted.
  - Add your files to the knowledge base: click the + icon > Upload Directory, then select the directory you created. All files will be added automatically.
  - When chatting, use the '#' sign to reference a KB

4. P4 as MCP Tool (UI Automation)

Implementation:

Create JSON coordinate mapping:

{ "browser_close_last_tab": {"x": 1200, "y": 50}, "powerpoint_new_slide": {"x": 100, "y": 200}, "file_save": {"x": 50, "y": 30} }

Expose as MCP tool:

ui_click(element_name: str) -> success/failure ui_type(element_name: str, text: str) -> success/failure

Required Demo: Closing the last tab in browser using coordinate automation

5. Web Search

Enable in Open WebUI settings (built-in capability)

Submission Guidelines

Submit a zip file with:

Source Code

- MCP server implementation for P1
- P4 coordinate automation tool code
- JSON coordinate mapping file

Demo Video

- Show demo of reimplemented P1 and P4
- **Must include:** Browser tab closing demo with P4
- Show successes AND failures
- Explain component interactions

Report

- **Architecture:** Component structure and communication
- **Performance:** Latency measurements and optimizations
- **Challenges & Limitations:** Issues encountered and solutions