CS 350 - Operating Systems

Fall 2025

LLM-Assisted System Monitor via FastMCP

Project Part 1 - Report

Group Members:
Nermin Tunçbilek
Menekşe Uzunçelebi
Müjgan Selen Karakaş

Introduction

Modern operating systems are capable of handling hundreds of processes at the same time, these processes using different amounts of CPU and memory. However, most users don't have an easy or natural way to understand what's happening on their computers or to manage these processes beyond using basic system monitors.

This project aims to develop an **LLM Assisted System Monitor** that lets users query their system in natural language; for example, by asking "Which process is using the most CPU right now?" and receive meaningful, human readable responses.

Using FastMCP (Model Context Protocol) as the link between the LLM client and a local system data server, the proposed system integrates AI with traditional OS process management.

The system will collect real time information such as CPU, RAM, and disk usage using the psutil Python library, then return the data to the LLM for interpretation. The LLM will summarize the result conversationally, for instance, "Your browser is consuming 45% CPU; would you like me to terminate it?"

In addition, the system will demonstrate LLM Cloud data integration by periodically storing collected performance data in a cloud database (e.g., Firebase or MongoDB Atlas) and retrieving it on demand. This allows users to ask questions such as "Show my system performance trend this week" and receive insights based on historical data.

This project brings together fundamental operating system concepts like processes, memory management, and CPU scheduling with modern AI tools to make system monitoring smarter and easier to interact with.

Tools and Technologies

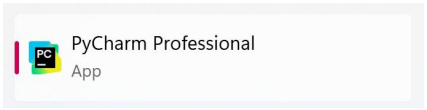
The development process will utilize:

- Python – used for both system monitoring and API integration.
- FastMCP – to establish the communication layer between the LLM and the system environment.
- psutil – to access CPU, memory, disk, and process details from the operating system.
- OpenAI API – for natural language understanding and response generation.
- Cursor IDE – As the LLM client environment to integrate the OpenAI API with the MCP server.
- Cloud Database (e.g., Firebase or MongoDB Atlas) – to store and retrieve system performance logs for long-term analytics and historical insights.
- Visual Studio Code / PyCharm – as the primary development environment, IDEs.

These tools were chosen because they allow efficient integration between system-level data and AI models, making it possible to monitor resources while maintaining natural, conversational interaction.

Software Setup

GitHub page: https://github.com/mujganselen/AssistedSystemMonitor




```
PS C:\Users\Mujgan> py --version
Python 3.14.0
```

```
PS C:\Users\Mujgan> py -m pip show fastmcp
Name: fastmcp
Version: 2.13.0.2
```

```
PS C:\Users\Mujgan> py -m pip show psutil
Name: psutil
Version: 7.1.3
```

```
PS C:\Users\Mujgan> py -m pip show openai
Name: openai
Version: 2.7.1
```

```
PS C:\Users\Mujgan> py -m pip show pymongo
Name: pymongo
Version: 3.12.0
```

Team Members and Responsibilities

Nermin Tunçbilek: Develop the FastMCP server and tool functions that read historical data from the cloud database and real time data from psutil

Müjgan Selen Karakaş: Configure cloud database, upload OS logs, retrieve historical data.

Menekşe Uzunçelebi: Configure the Cursor IDE with the OpenAI API key, integrating the client with the FastMCP server, prepare demo

All members will collaborate on integration, testing, and performance evaluation.

System Design and Algorithms

The system consists of four main components:

FastMCP Server: This part connects the operating system with the LLM. It uses the psutil library to collect data such as CPU usage, memory consumption, disk space, and running processes.

Process Control Module: Allows the user to perform actions like terminating, suspending, or resuming a process safely.

Cloud Logging Module : Periodically stores system performance data in a cloud database and retrieves historical logs.

LLM Client: Handles the user's natural language queries, sends structured commands to the FastMCP server, and generates human readable explanations based on the data it receives.

Workflow:

1. The user asks a question such as "Which process is using the most CPU?" or "Show the last 1 hour trend"
2. The LLM understands the question and sends a request through FastMCP.
3. FastMCP uses psutil to collect the requested data from the OS if the query is about real time data, retrieved from cloud logs if historical data.
4. The LLM processes the data and returns a natural language explanation to the user.

Future enhancements may include visualization tools (e.g., graphs for CPU/RAM usage) and voice-based interactions.

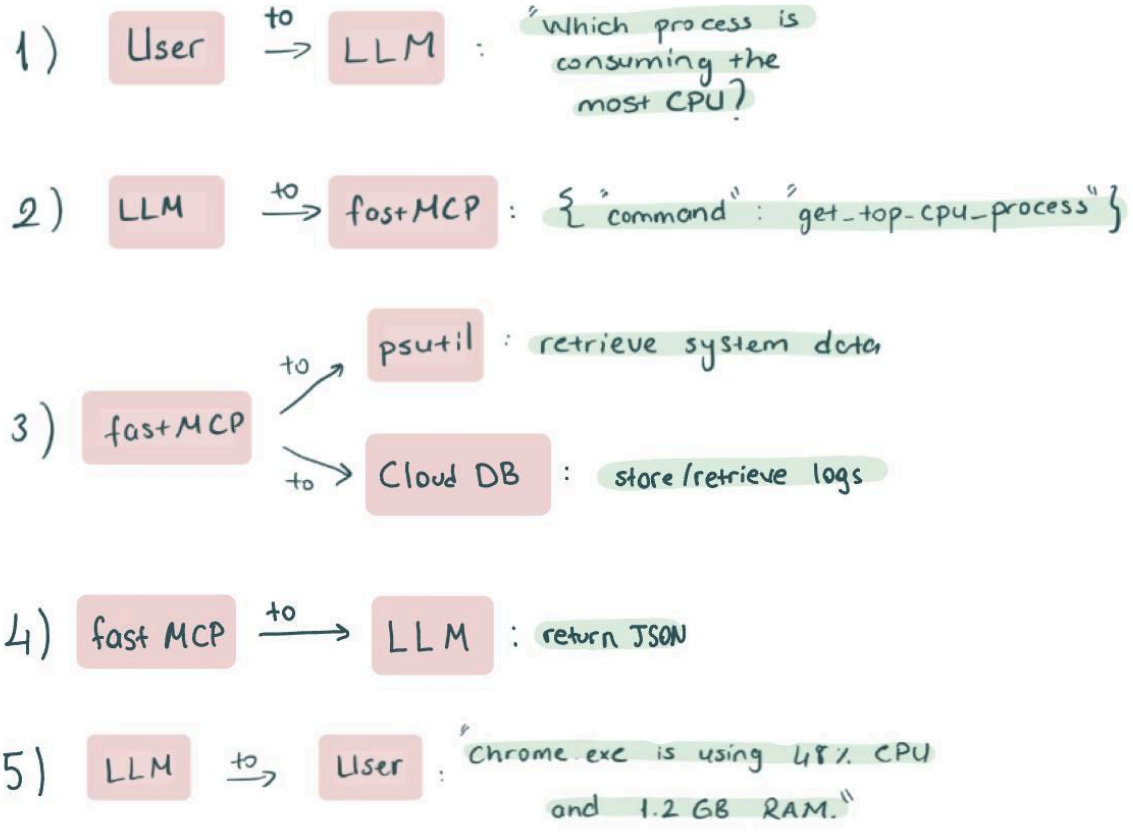
Expected Results and Analysis

The final system is expected to:

- Monitor processes and resource usage in real time.
- Interpret technical data into natural, easy to understand explanations.
- Execute basic process control commands safely.
- Demonstrate a working FastMCP and LLM integration prototype.
- Store and retrieve system performance logs through a cloud database for historical analysis and long term monitoring.

Through this project, we aim to show how LLM driven natural language interfaces can simplify complex operating system monitoring tasks and bridge the gap between user interactions and low level system operations.

Our analysis will focus on the system's accuracy in responding to user queries, the latency of data retrieval through FastMCP and cloud storage, and the reliability of process control commands. Overall, this project intends to demonstrate how AI based interfaces can improve the interaction between humans and computer in operating systems.



Sample Workflow

References

Psutil Documentation. (2025). https://psutil.readthedocs.io

OpenAI Developer Documentation. (2025). https://platform.openai.com/docs

Model Context Protocol (MCP) Specification, OpenAI, 2024.

Tanenbaum, A. S., & Bos, H. (2015). Modern Operating Systems (4th ed.). Pearson.