

# Machine Learning Engineering Program

Module1: MCP Homework and Local LLM Implementation  
with Ollama & LangChain

Sophie Gao

# Part 1 Using MCP to built Agent-like work-flow.

## Task 1.1 MCP + Claude = Browser Automation

### 1. Brave Search Plugin

Task: Search for the latest AI paper publication platforms

Status: Configuration issue - API authentication error

Outcome: Provided alternative information about arXiv.org, Papers with Code, and OpenReview.

### 2. GitHub Plugin

Task: Connect to GitHub and list 5 latest commits from repository SophieG22/Dashboards

Status: Successfully completed

Results:

- Retrieved 3 total commits (all from November 16, 2024)
- Latest commit: Add files via upload (052ffc6e)
- All commits were cryptographically verified

### 3. Puppeteer Plugin

Task: Navigate to <https://www.inference.ai/> and capture a full-page screenshot

Status: Successfully completed

Details:

- Screenshot saved as example.png
- Captured at 800x600 pixels
- Page shows: Fractionalized GPUs service with dark theme

### 4. Filesystem Plugin

Task: Create folder named mcp\_test on Desktop and add hello.txt file

Status: Successfully completed

Actions:

- Created folder: C:\Users\FeiFei\Desktop\mcp\_test
- Created file: hello.txt with content 'Hello MCP!'
- Both operations completed without errors

### 5. Sequential Thinking Plugin

Task: Create a step-by-step plan for preparing for a technical interview

Status: Successfully completed

Output: Comprehensive 17-step guide covering:

- Research and understanding (interview format, CS fundamentals)
- Active practice (coding problems, system design, domain knowledge)
- Communication, health management, and follow-up strategies

### 6. Notion Plugin

Task: Create new page with summary of completed tasks

Status: Successfully completed

Actions:

- Created page in database: MCP Automation Test - Summary of Completed Tasks
- Added comprehensive summary of all 6 plugin tests with details
- Structured content with headings, paragraphs, and bullet points

# Part 1 Using MCP to built Agent-like work-flow.

## Task 1.1 MCP + Claude = Browser Automation

### Summary Statistics

Total plugins tested: 6

Successfully completed: 5

Partial/Issues: 1 (Brave Search - API configuration)

Overall success rate: 83%

---

Test completed on October 24, 2025

# Advanced Task

**SG** Run the Puppeteer scraper on <https://datatables.net.cn/examples/styling/display.html>, save the table as saved as datatable.png, upload it to Notion, and summarize it in documentation

I'll help you scrape the DataTables example page, save the table as an image, upload it to Notion, and create documentation. Let me break this down into steps:

1. First, I'll navigate to the page and take a screenshot of the table
2. Then search for a suitable Notion page to upload to
3. Upload the image and create documentation

Let's start:

- P** Puppeteer navigate
- P** Puppeteer screenshot

Great! I've captured the table. Now let me search for Notion pages where I can upload this:

# Advanced Task

Perfect! I have successfully completed all the tasks:

## Summary

✅ **Task Completed Successfully!**

Here's what I did:

1. **Scraped the DataTables page** - Used Puppeteer to navigate to <https://datatables.net.cn/examples/styling/display.html>
2. **Captured the table screenshot** - Saved a 1200x800 pixel screenshot of the employee data table as `datatable.png`
3. **Uploaded to Notion** - Added comprehensive documentation to your Notion "Habit tracker" page with:
  - Detailed overview of the DataTable
  - Table structure breakdown (all 6 columns explained in Chinese and English)
  - Key features highlighting sortable columns, clean design, and responsive layout

# Advanced Task

- Sample data insights including highest salary, youngest/oldest employees, and office distribution
- Source link and metadata

#### 4. **Created Documentation** - The documentation includes:

- Full analysis of the employee information system
- Column descriptions (Name, Position, Office, Age, Start Date, Salary)
- Notable insights from the visible data
- Reference to the saved screenshot file

You can view the updated Notion page here: <https://www.notion.so/Habit-tracker-29647cd79629803fb1d4ee690a8b66b1>

The screenshot `datatable.png` has been saved and captures the Chinese language employee table showing 10 employees with their complete information across all columns!

# Part 2 Play with Ollama

## Task 2.1: Install Ollama and run LLMs locally

```
C:\Users\FeiFei>ollama run llama2
>>> where's the capital of canada?

The capital of Canada is Ottawa.

>>> Send a message (/? for help)
```



# Part 2 Play with Ollama

## Task 2.2: Using Ollama to call OpenAI API

```
from openai import OpenAI

client = OpenAI(
    base_url = 'http://localhost:11434/v1',
    api_key='ollama', # required, but unused
)

response = client.chat.completions.create(
    model="llama2",
    messages=[
        {"role": "system", "content": "You are a helpful assistant."},
        {"role": "user", "content": "Who won the world series in 2020?"},
        {"role": "assistant", "content": "The LA Dodgers won in 2020."},
        {"role": "user", "content": "Where was it played?"}
    ]
)

print(response.choices[0].message.content)
```

The 2020 World Series was played at various locations, including:

- \* Dodger Stadium in Los Angeles, California (home of the LA Dodgers)
- \* Fenway Park in Boston, Massachusetts (home of the Boston Red Sox)
- \* Globe Life Field in Arlington, Texas (home of the Texas Rangers)
- \* Nationals Park in Washington, D.C. (home of the Washington Nationals)
- \* Oracle Park in San Francisco, California (home of the San Francisco Giants)

The series was played from October 21 to November 5, 2020, with the LA Dodgers winning in 6 games over the Boston Red Sox.



# Part 3 Combining LangChain with Ollama's local LLMs

## Task 3.1 Reproduce practice in lecture using LCEL

```
# 4. Chain the components together using LCEL
chain = (
    # LCEL syntax: use the pipe operator | to connect each step
    {"topic": RunnablePassthrough()} # Accept user input
    | prompt                          # Transform it into a prompt message
    | model                           # Call the model
    | StrOutputParser()               # Parse the output as a string
)

# 5. Execute
result = chain.invoke("Germany")
print("User prompt: 'What is the capital of Germany?'")
print("Model answer:", result)
```

```
User prompt: 'What is the capital of Germany?'
Model answer: The capital of Germany is Berlin.
```

# Advance Work

Use **Direct** mode to call Ollama via the OpenAI SDK, or **LangChain** mode to route through LangChain.

Make sure Ollama is running ( `ollama serve` ) and you have pulled a model (e.g. `ollama pull llama3` ).

User input

where is the capital of canada

Model name (Ollama)

llama3

☐ Translate first (proxy translator)

Source language (when translate)

Chinese

Target language (when translate)

English

Run — Direct Ollama

Run — LangChain (optional)

Model output

The capital of Canada is Ottawa!

Raw debug output (optional)