

Class 14: Model Context Protocol (MCP)

- **Review:** AGENTS.md
- **Lecture:** Model Context Protocol (MCP)
- **Studio:** Homework 4

Copyright 2026, Kyle J. Harms

Review: Efficient Prompt Engineering

- **Specific prompts:** Clearly define the task and desired output.
- **Descriptive prompts:** Provide context and details to guide the model's response.
- **Contextual prompts:** Include relevant information or background to help the model understand the task. (Avoid overloading context; don't dump the entire codebase into the prompt.)
- **Example-based prompts:** Provide examples of the desired output to guide the model's response.
- **Constraint-based prompts:** Specify any constraints or limitations on the output (e.g., length, format, style, etc.)

Review: AGENTS.md

AGENTS.md provides a dedicated set of instructions for the LLM to follow when generating code.

Best Practices:

- Put commands at the top. (e.g. `npm run build`)
- Provide code examples over explanations
- Provide clear boundaries (e.g. “What files can the AI touch and not touch?”)
- Provide specifics about your tech stack. (e.g. “What version of Vue.js?”)

Model Context Protocol (MCP)

Model Context Protocol (MCP)

MCP is an open protocol that lets AI agents connect to external servers that provide tools (actions the agent can call), resources (data/docs), and prompts.

Example: a docs server lets Copilot look up current DaisyUI/Vue docs instead of guessing.

MCP Capabilities

Tools: Executable functions/actions that the AI agent can call.

Resources: Data or documentation that the AI agent can access.

Prompts: Predefined templates or workflows that guide the interactions between the user and the AI model.

MCP: Tools

Executable functions/actions that the AI agent can call.

```
function getWeather(location) {  
    // Connect to weather API and fetch data  
    return {  
        temperature: 72,  
        conditions: 'Sunny',  
        humidity: 45  
    };  
}
```

Use: #NAME_OF_TOOL PROMPT

Example:

```
#getWeather "New York, NY"
```

MCP: Resources

Data or documentation that the AI agent can access.

```
function readFile(filePath) {  
  // Using fs.readFile to read file contents  
  const fs = require('fs');  
  return new Promise((resolve, reject) => {  
    fs.readFile(filePath, 'utf8', (err, data) => {  
      if (err) {  
        reject(err);  
        return;  
      }  
      resolve(data);  
    });  
  });  
}
```

Use: *Add context, **MCP Resources...**, select the resource*

MCP: Prompts

Predefined templates or workflows that guide the interactions between the user and the AI model.

```
function codeReview(code, language) {  
  return [  
    {  
      role: 'system',  
      content: `You are a code reviewer examining  
${language} code. Provide a detailed review  
highlighting best practices, potential issues,  
and suggestions for improvement.`  
    },  
    {  
      role: 'user',  
      content: `Please review this ${language}  
code:\n\n\`${code}\`${language}\n${code}\n\`${code}\`  
    }  
  ];  
}
```

Use: `/MCP_PROMPT_NAME PROMPT`

Example:

```
/mcp.example.codeReview  
"function add(a, b) { return a + b; }"  
"JavaScript"
```

`.mcp.json`

Add MCP servers to your project by creating a `.mcp.json` file in the root of your project.

```
{
  "mcpServers": {
    "foo": {
      "type": "http",
      "url": "https://foo.com/mcp"
    }
  }
}
```

Gotcha: You may need to start/enable the server in your code editor.

Class Activity: `.mcp.json`

1. Create a `.mcp.json` file for your Homework 4 project.
2. Add the DaisyUI MCP server to your `.mcp.json` file. (`mcpServers` object)

```
{  
  "mcpServers": {  
    "daisyUI": {  
      "type": "sse",  
      "url": "https://gitmcp.io/saadeghi/daisyui"  
    }  
  }  
}
```

Class Activity: Explore MCP

1. Explore the tools, resources, and prompts available.
2. Use the DaisyUI MCP server to look up the docs for the rating component.
3. Use the rating component docs to create a Vue.js rating component of the DaisyUI rating component.

Homework 4

A **task specific** (or domain/discipline specific) chatbot.

Examples:

- A [presidential poem](#) creator
- DaisyUI component to Vue.js component converter.
- A validating “customer service” chatbot for a specific product or service.
- A trivia bot that only answers questions about a specific topic (e.g. history, science, etc.)

Activity: Task Specific Chat Bot

Complete your handout to brainstorm ideas for your task/domain/discipline specific chat bot.

Reminder

Prompt engineering, especially context, can overcome many limitations of an LLM model, including **tiny** LLMs.

Yes, it's true that the model we're using doesn't "know" anything after 2023. But it still follows instructions pretty well. Provide it with the right context and instructions, and see if it can generate what you need.

Studio: Homework 4

Work on your task specific (or domain/discipline specific) chatbot.