MCP – Model Context Protocol

LLM can answer based on what it knows already but not beyond its training. Say if you ask “What’s Java?” LLM can provide the answer. If you ask such a question where LLM has to do some extra work to provide the answer, you need to provide supportive tools like DuckDuckGoSearchRun that LLM can call otherwise LLM will hallucinate. LLM is a separate part, tools are a separate part.

Example code snippet
```
   @GetMapping("/api/{message}")
   public String home(@PathVariable String message) {
      ChatResponse response = chatClient
            .prompt(message)
            .tools(dateTimeTool, newsTool)
            .call()
            .chatResponse();  //.content();
```

You want something and Agents do it for you and thats the functionality of Agents. Imagine the tools we have created so far are Agents because we want something and they are executing on your behalf, similar to Agents. Lets say we want to push a project to GitHub, the changes need to be added, committed with a good messaged then pushed into Git. If I change the Copilot mode to “Agent”, it can perform tasks on IDE and that means it has control on IDE. However, if I ask Copilot to push code onto Git. Can it do it? To do this, GitHub has to give that power. Say if GitHub says I want to give that capability to the developers. Ultimately, GitHub is a platform for developers and they should do something to achieve this. What if GitHub on their server have some tools pre-created for creating a repository, reading a repository, comparing the conflicts, comparing the branches, creating branches etc. In fact, GitHub already have their own tools that can connect with your LLM. You can write your own code, which can talk to the tools of GitHub so you can make that integration. Next you want to integrate with the chat platform of your company, could be Slack, Zoho etc. As soon as you push the code to GitHub, you want to notify on Slack channel. Slack needs to create some tools to achieve that. Your job is to write some Client or code that will interact with those tools. If you want to interact with remote tools, they have to be exposed via Rest API then you can call them. Slack needs to have some in-built tools and expose them to client like us. Similarly, GitHub also needs to create some tools and expose them via Rest APIs. When I work as a Developer, I interact with multiple services like GitHub, Slack, Jira etc. Even Jira needs to expose some tools in order to achieve that. When that’s done, we could even use LLM/Copilot to say “Create a new issue on Jira, Create a branch on GitHub, Post on Slack channel”. For one prompt, you are interacting with multiple services/tools. We have to write tools on our machine (Client) to interact with the tools of Slack, Jira, GitHub etc. But there is one problem, to interact with GitHub or Jira, we got to write a separate custom code. In the future, if they change any of the feature, we got to modify our code also accordingly. If they move to some other server or change AI service, we got to change in our end also. How do we solve this problem? What if all these services follow a particular standard like Rest API or JSON? First we had Nokia phone with a small pin, then micro-USBs, iPhone lightning port but now we have USB-C in all phones. If you have a standard pin, then it is easier to communicate. Similarly, what if the Servers that are exposing their Endpoints follow the same standards. And that’s possible with the help of **Model-Context Protocol (MCP)**.
https://modelcontextprotocol.io/docs/getting-started/intro

What is the Model Context Protocol (MCP)?
MCP (Model Context Protocol) is an open-source standard for connecting AI applications to external systems. Using MCP, AI applications like Claude or ChatGPT can connect to data sources (e.g. local

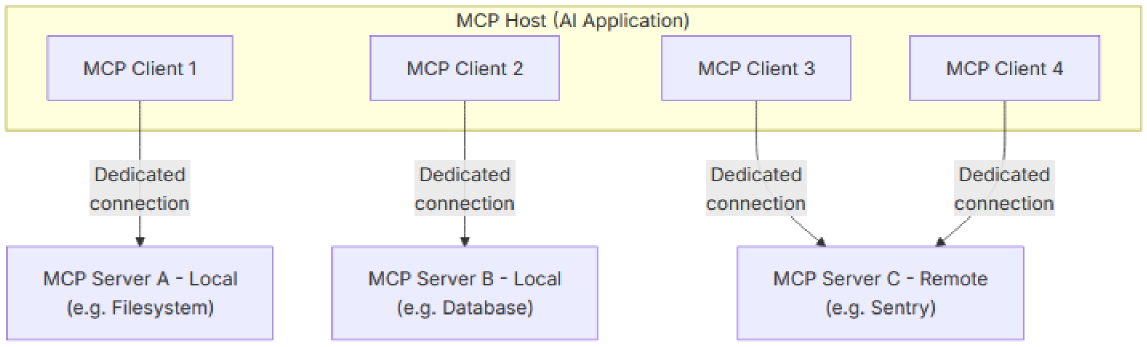
files, databases), tools (e.g. search engines, calculators) and workflows (e.g. specialized prompts)—enabling them to access key information and perform tasks. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect electronic devices, MCP provides a standardized way to connect AI applications to external systems.

MCP saves time to connect to external services. Who has built MCP? Anthropic. All major AI providers are using MCP now. Using MCP, ChatGPT or Claude can connect to data sources. Different external services like GitHub or Jira build their own MCP servers and what we are going to build is MCP clients. We have to build a client, connect with a server our job is over. Think of MCP as a USB-C port for all the AI applications.

We are going to use SpringAI for MCP
https://docs.spring.io/spring-ai/reference/api/mcp/mcp-overview.html

SpringAI provides abstraction over Java SDK, similarly, MCP provides abstraction for tool calling. In SpringAI, we can connect with different AI models from OpenAI to Gemini (Eg: spring.ai.openai.api-key=Your_API_Key, spring.ai.openai.chat.options.model=gpt-4o). Spring can connect with different databases from Oracle to MySQL etc. GitHub first needs to create an MCP server then you can create an MCP client.

MCP Host: Machine where you are writing your AI code. Our machine where we have written code in Intelij Idea is our MCP Host
MCP Client: We can create an MCP client on our MCP host
MCP Server: Server that provides those features. Say create a repository on GitHub thats one feature this Server will have. It can have multiple tools or features

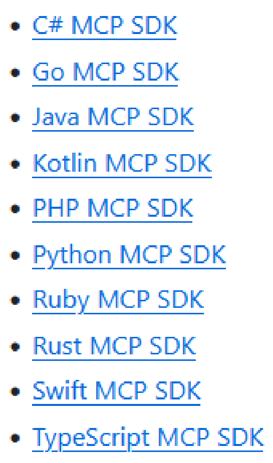

MCP Host (AI Application)
MCP Client 1
MCP Client 2
MCP Client 3
MCP Client 4
Dedicated connection
Dedicated connection
Dedicated connection
Dedicated connection
MCP Server A - Local (e.g. Filesystem)
MCP Server B - Local (e.g. Database)
MCP Server C - Remote (e.g. Sentry)

This machine is an MCP Host and we are creating an AI application. In my machine, I can create multiple MCP Clients. One MCP Client can connect to one MCP Server. Say one Server is for the Filesystem, one for Database and one for Remote service like Sentry, GitHub. We can run MCP Server running on the local machine or remote machine. For example, Sentry runs on remote machine, GitHub service runs on the remote machine. Today we will run MCP Server running on the local machine. If you want to call MCP Server, we have to call MCP Client. MCP is not a framework, it is a protocol. When you write an MCP Server, you are not doing anything extra, you are creating a tool which you are writing in the language of MCP. Different protocols use different languages. TCP have their own language to interact between two machines. HTTP has its own language to interact. When you create a tool on the Server, we (Client) have to just adopt the MCP language.

https://github.com/modelcontextprotocol

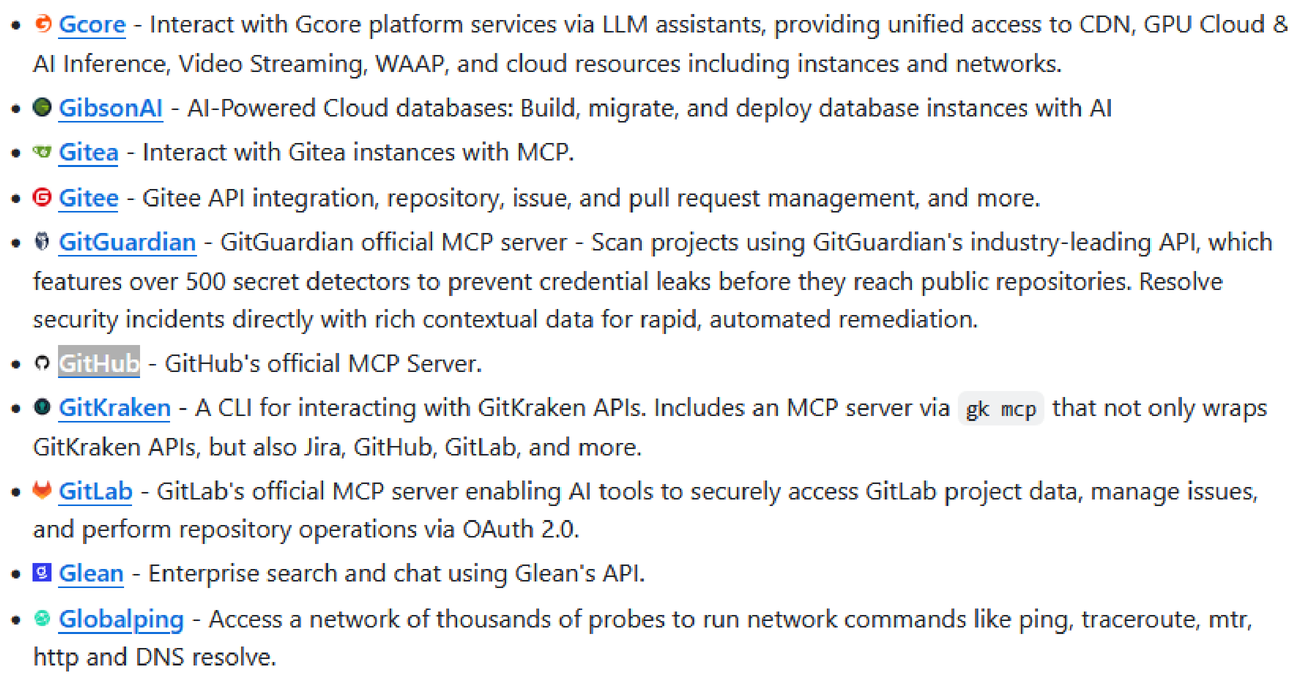
We have different SDKs available. Java SDK is basically SpringAI itself. If you want to work with Java and MCP, we have just one option, SpringAI.

If you go to Servers: https://github.com/modelcontextprotocol/servers
It shows all our Servers available



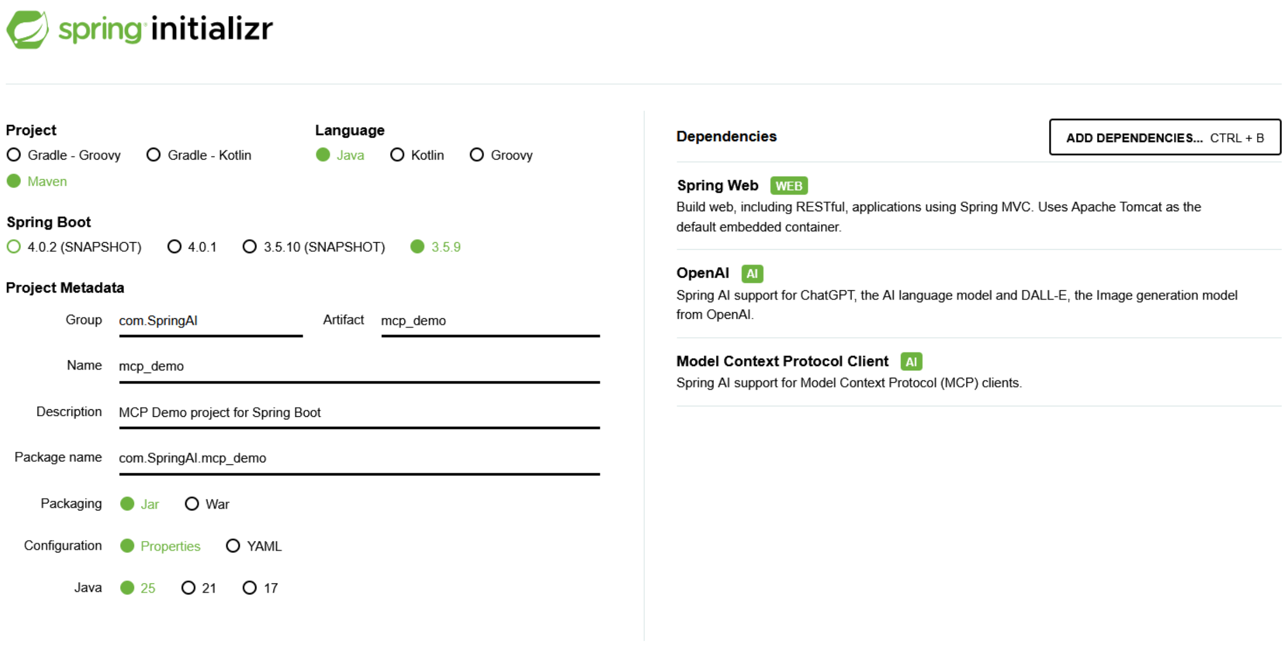
- C# MCP SDK
- Go MCP SDK
- Java MCP SDK
- Kotlin MCP SDK
- PHP MCP SDK
- Python MCP SDK
- Ruby MCP SDK
- Rust MCP SDK
- Swift MCP SDK
- TypeScript MCP SDK

There are so many MCP servers already created, we can call them and use them



- Gcore - Interact with Gcore platform services via LLM assistants, providing unified access to CDN, GPU Cloud & AI Inference, Video Streaming, WAAP, and cloud resources including instances and networks.
- GibsonAI - AI-Powered Cloud databases: Build, migrate, and deploy database instances with AI
- Gitea - Interact with Gitea instances with MCP.
- Gitee - Gitee API integration, repository, issue, and pull request management, and more.
- GitGuardian - GitGuardian official MCP server - Scan projects using GitGuardian's industry-leading API, which features over 500 secret detectors to prevent credential leaks before they reach public repositories. Resolve security incidents directly with rich contextual data for rapid, automated remediation.
- GitHub - GitHub's official MCP Server.
- GitKraken - A CLI for interacting with GitKraken APIs. Includes an MCP server via `gk mcp` that not only wraps GitKraken APIs, but also Jira, GitHub, GitLab, and more.
- GitLab - GitLab's official MCP server enabling AI tools to securely access GitLab project data, manage issues, and perform repository operations via OAuth 2.0.
- Glean - Enterprise search and chat using Glean's API.
- Globalping - Access a network of thousands of probes to run network commands like ping, traceroute, mtr, http and DNS resolve.

Create a new Spring project: https://start.spring.io/
Dependencies: Spring Web, OpenAI, MCP Client. We will use the existing MCP Server


spring initializr

Project
○ Gradle - Groovy ○ Gradle - Kotlin [✓ Maven]

Language
[✓ Java] ○ Kotlin ○ Groovy

Spring Boot
○ 4.0.2 (SNAPSHOT) ○ 4.0.1 ○ 3.5.10 (SNAPSHOT) [✓ 3.5.9]

Project Metadata

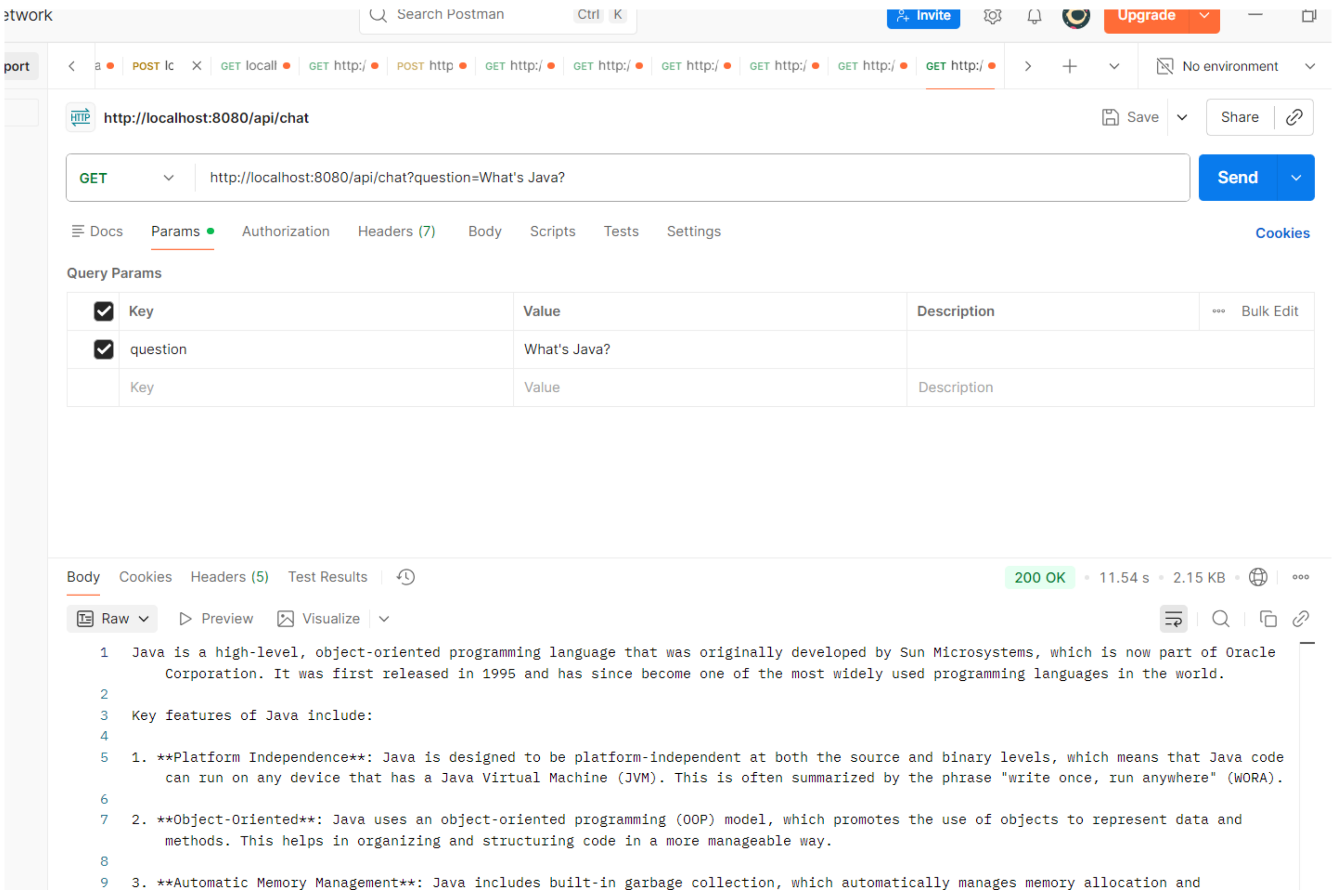
Group	com.SpringAI
Artifact	mcp_demo
Name	mcp_demo
Description	MCP Demo project for Spring Boot
Package name	com.SpringAI.mcp_demo
Packaging	[✓ Jar] ○ War
Configuration	[✓ Properties] ○ YAML
Java	[✓ 25] ○ 21 ○ 17

Dependencies ADD DEPENDENCIES... CTRL + B

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

OpenAI AI
Spring AI support for ChatGPT, the AI language model and DALL-E, the Image generation model from OpenAI.

Model Context Protocol Client AI
Spring AI support for Model Context Protocol (MCP) clients.



http://localhost:8080/api/chat

GET http://localhost:8080/api/chat?question=What's Java?

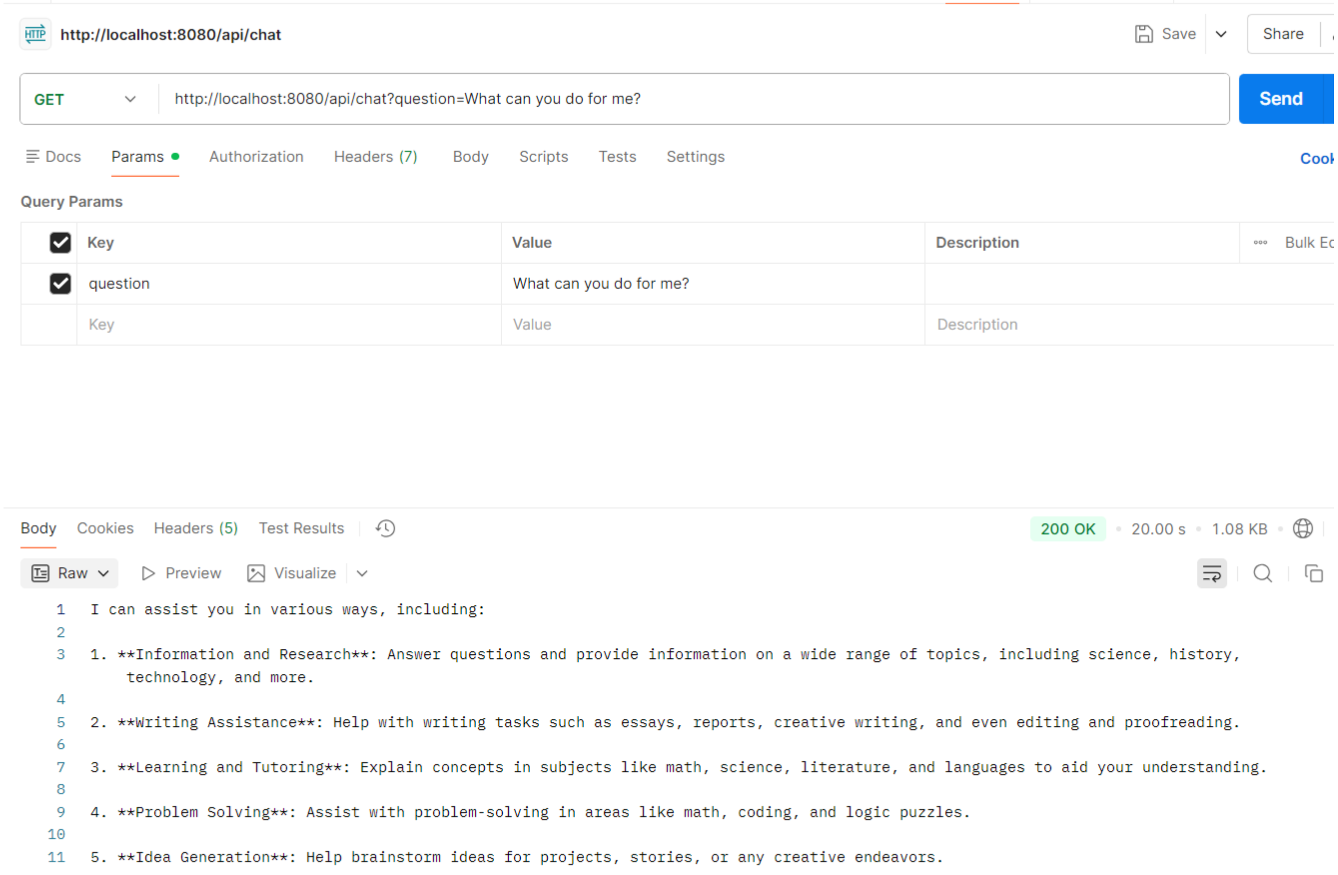
Query Params

	Key	Value	Description
✓	question	What's Java?	
	Key	Value	Description

Body Cookies Headers (5) Test Results 200 OK • 11.54 s • 2.15 KB

```
Java is a high-level, object-oriented programming language that was originally developed by Sun Microsystems, which is now part of Oracle
    Corporation. It was first released in 1995 and has since become one of the most widely used programming languages in the world.

Key features of Java include:

1. **Platform Independence**: Java is designed to be platform-independent at both the source and binary levels, which means that Java code
    can run on any device that has a Java Virtual Machine (JVM). This is often summarized by the phrase "write once, run anywhere" (WORA).

2. **Object-Oriented**: Java uses an object-oriented programming (OOP) model, which promotes the use of objects to represent data and
    methods. This helps in organizing and structuring code in a more manageable way.

3. **Automatic Memory Management**: Java includes built-in garbage collection, which automatically manages memory allocation and
```

AI can do all these things but it cannot help me with for example, FileSystem: like create a folder

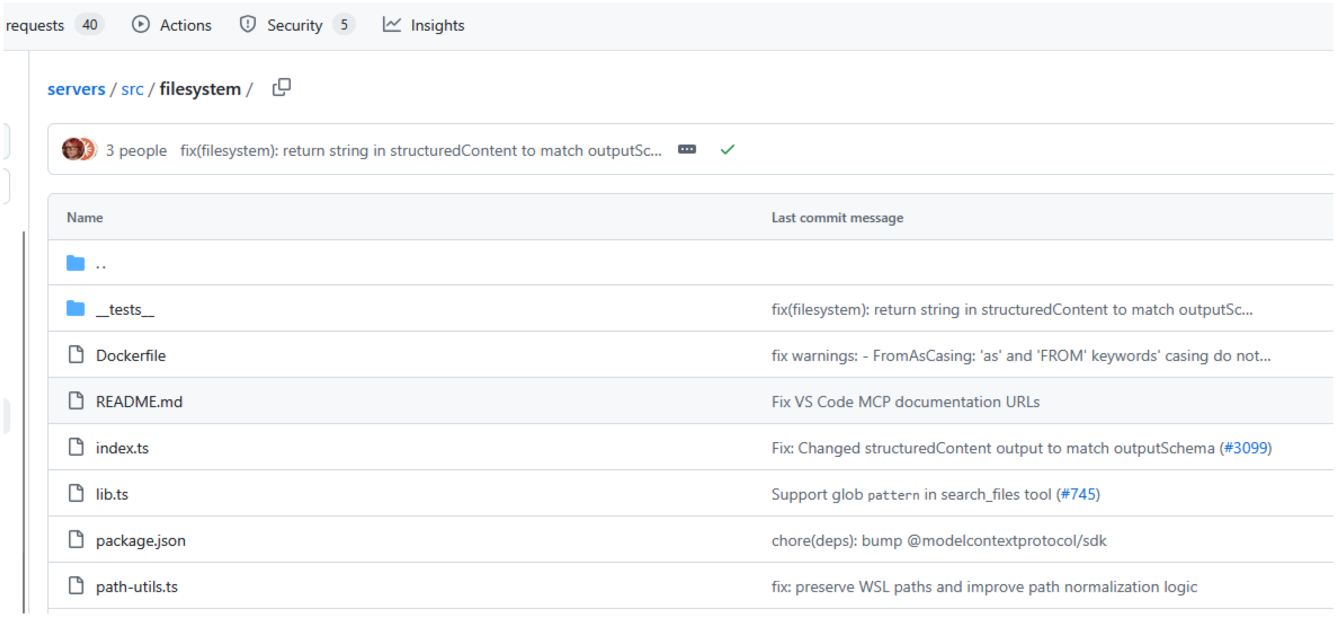


http://localhost:8080/api/chat?question=What can you do for me?

Key	Value	Description
question	What can you do for me?	
Key	Value	Description

```
I can assist you in various ways, including:

1. **Information and Research**: Answer questions and provide information on a wide range of topics, including science, history,
   technology, and more.

2. **Writing Assistance**: Help with writing tasks such as essays, reports, creative writing, and even editing and proofreading.

3. **Learning and Tutoring**: Explain concepts in subjects like math, science, literature, and languages to aid your understanding.

4. **Problem Solving**: Assist with problem-solving in areas like math, coding, and logic puzzles.

5. **Idea Generation**: Help brainstorm ideas for projects, stories, or any creative endeavors.
```

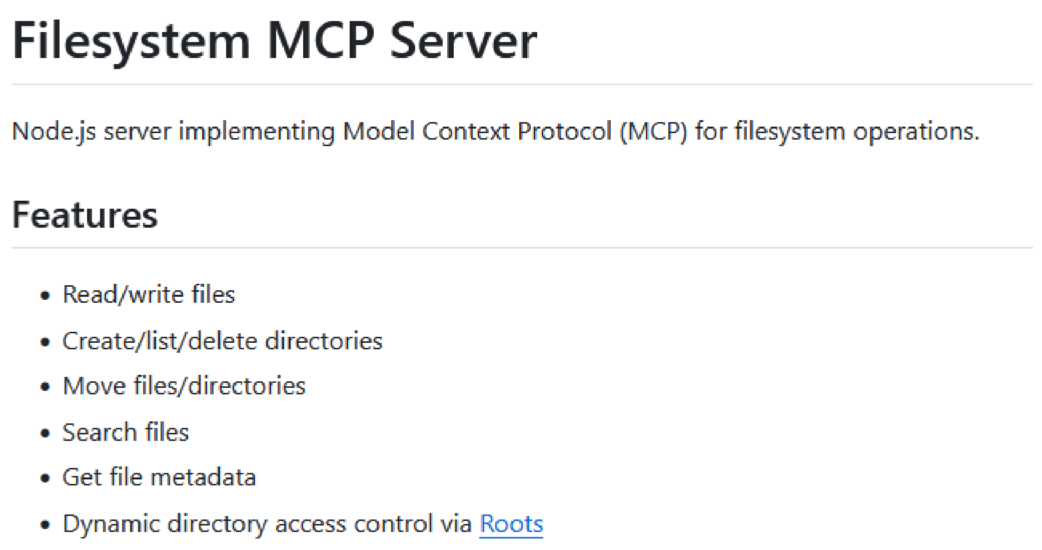
If you go to GitHub and filesystem
https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem



3 people fix(filesystem): return string in structuredContent to match outputSc...

Name	Last commit message
..	
__tests__	fix(filesystem): return string in structuredContent to match outputSc...
Dockerfile	fix warnings: - FromAsCasing: 'as' and 'FROM' keywords' casing do not...
README.md	Fix VS Code MCP documentation URLs
index.ts	Fix: Changed structuredContent output to match outputSchema (#3099)
lib.ts	Support glob `pattern` in search_files tool (#745)
package.json	chore(deps): bump @modelcontextprotocol/sdk
path-utils.ts	fix: preserve WSL paths and improve path normalization logic

We know how to call tools. Still instead of writing our own Tool codes and calling them, we can use MCP Server that was already created



Filesystem MCP Server

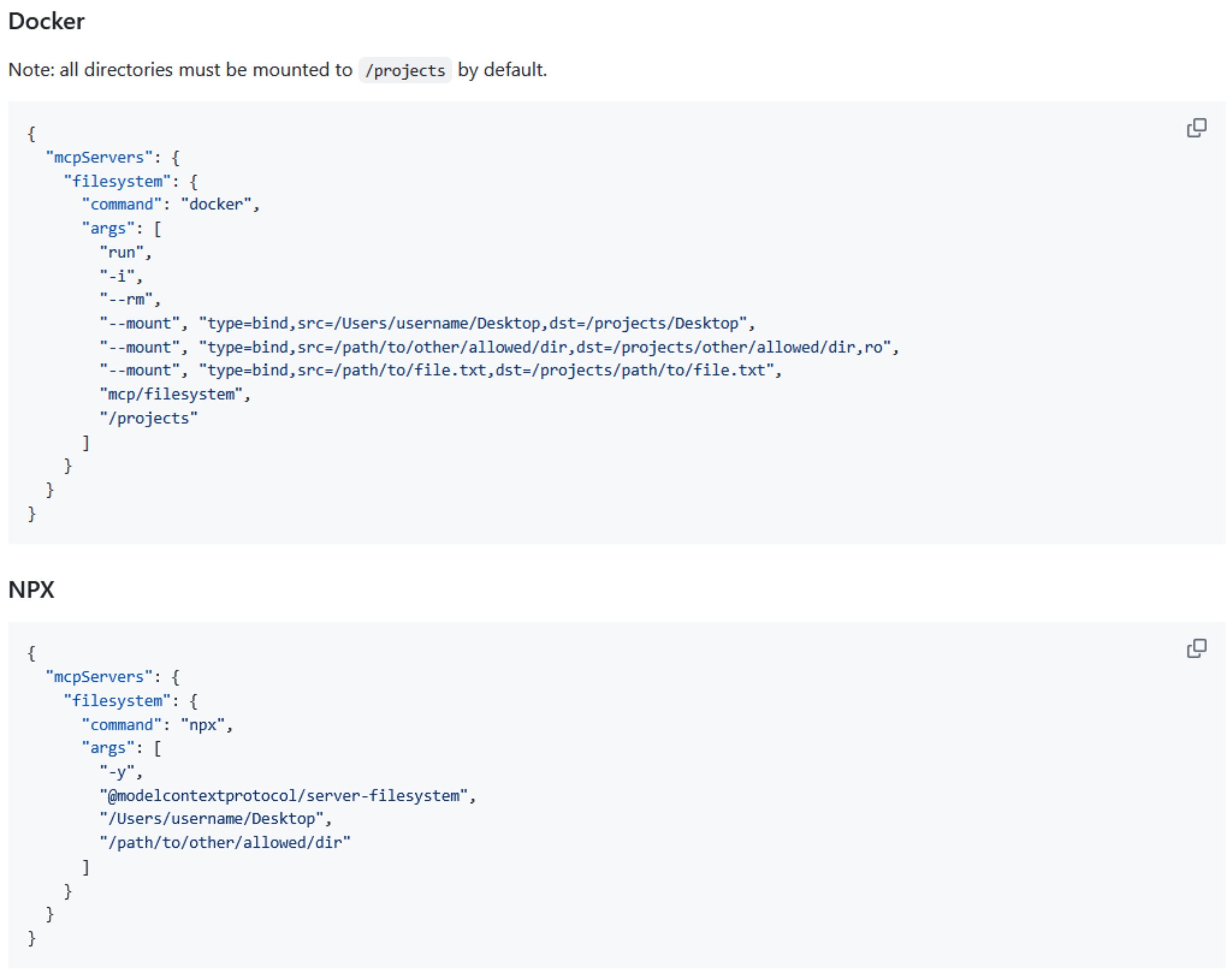
Node.js server implementing Model Context Protocol (MCP) for filesystem operations.

Features

- Read/write files
- Create/list/delete directories
- Move files/directories
- Search files
- Get file metadata
- Dynamic directory access control via Roots

mcp-server-filesystem /path/to/dir1 /path/to/dir2
Any MCP Server can either run on local machine or remote machine. For now, we are going to run on local

We can run with either Docker or NPX



Docker

Note: all directories must be mounted to `/projects` by default.

```
{
  "mcpServers": {
    "filesystem": {
      "command": "docker",
      "args": [
        "run",
        "-i",
        "--rm",
        "--mount", "type=bind,src=/Users/username/Desktop,dst=/projects/Desktop",
        "--mount", "type=bind,src=/path/to/other/allowed/dir,dst=/projects/other/allowed/dir,ro",
        "--mount", "type=bind,src=/path/to/file.txt,dst=/projects/path/to/file.txt",
        "mcp/filesystem",
        "/projects"
      ]
    }
  }
}
```

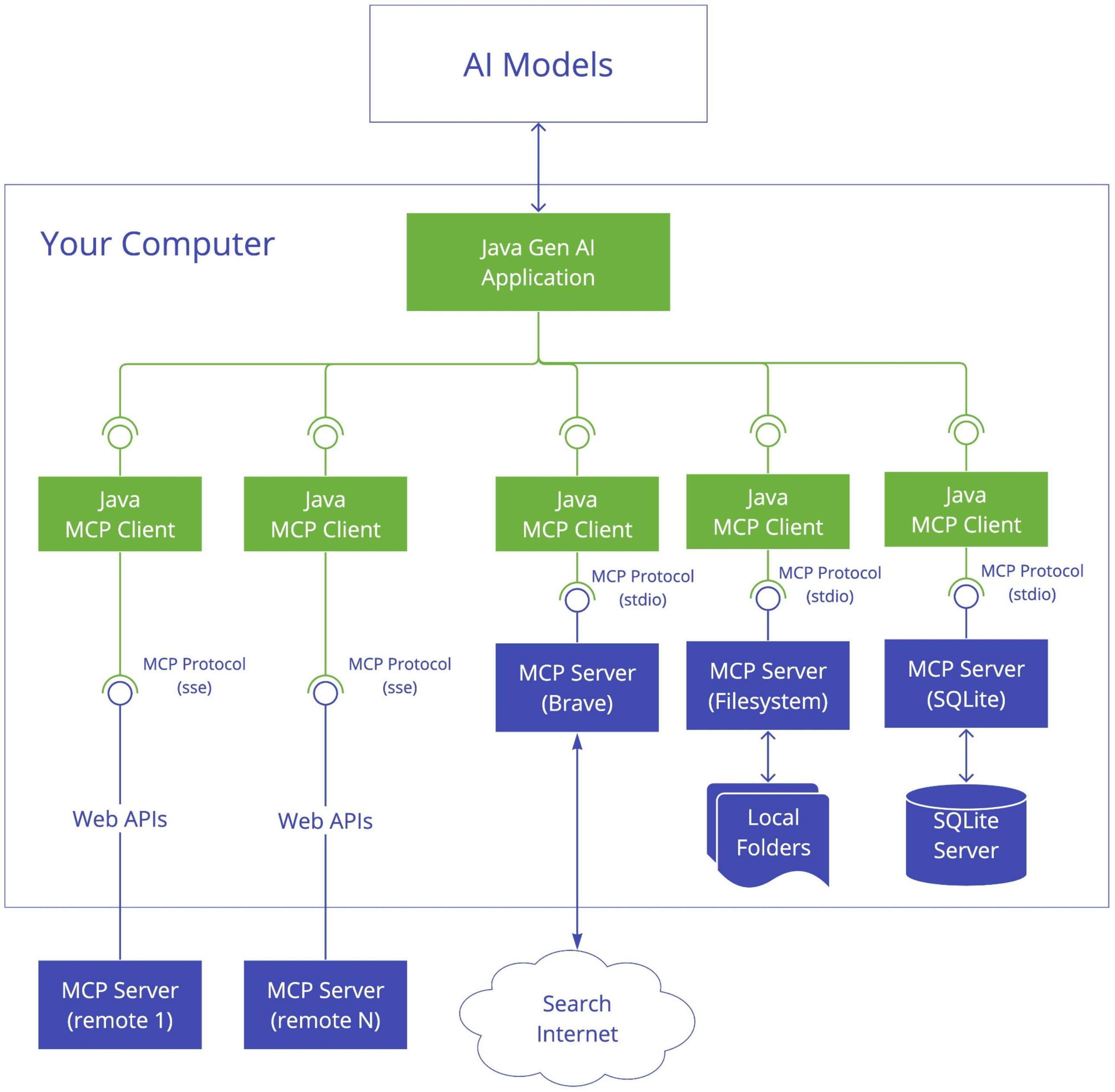
NPX

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/username/Desktop",
        "/path/to/other/allowed/dir"
      ]
    }
  }
}
```

To create MCP Client: **spring.ai.mcp.client.stdio.servers-configuration=classpath:mcp-servers.json**

we are using stdio because it is on local

https://docs.spring.io/spring-ai/reference/api/mcp/mcp-overview.html



For a Java client to connect to MCP server running on local, we got to use stdio
For Java client to connect to MCP server running on remote, we got to use sse
sse is deprecated, new one is Streamable-HTTP
Which MCP Server to connect and which features we want to expose, must be specified in mcp-servers.json

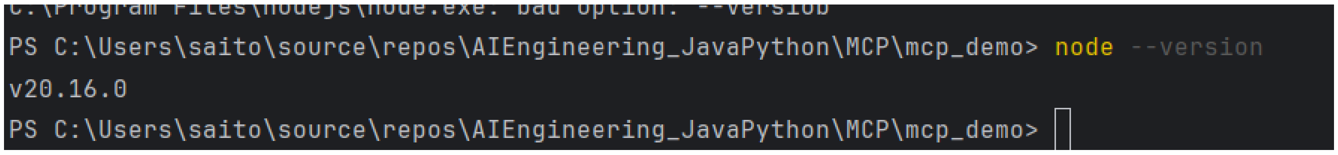
```
servers-configuration=classpath:mcp-servers.json
```

in the resources folder, lets create mcp-servers.json

I have copy-pasted the NPX code

```
{
  "mcpServers": {
    "filesystem": {
      "command": "npx",
      "args": [
        "-y",
        "@modelcontextprotocol/server-filesystem",
        "/Users/username/Desktop",
        "/path/to/other/allowed/dir"
      ]
    }
  }
}
```

Check Node is installed



```
C:\Program Files\nodejs\node.exe: bad option: --versiob
PS C:\Users\saito\source\repos\AIEngineering_JavaPython\MCP\mcp_demo> node --version
v20.16.0
PS C:\Users\saito\source\repos\AIEngineering_JavaPython\MCP\mcp_demo>
```

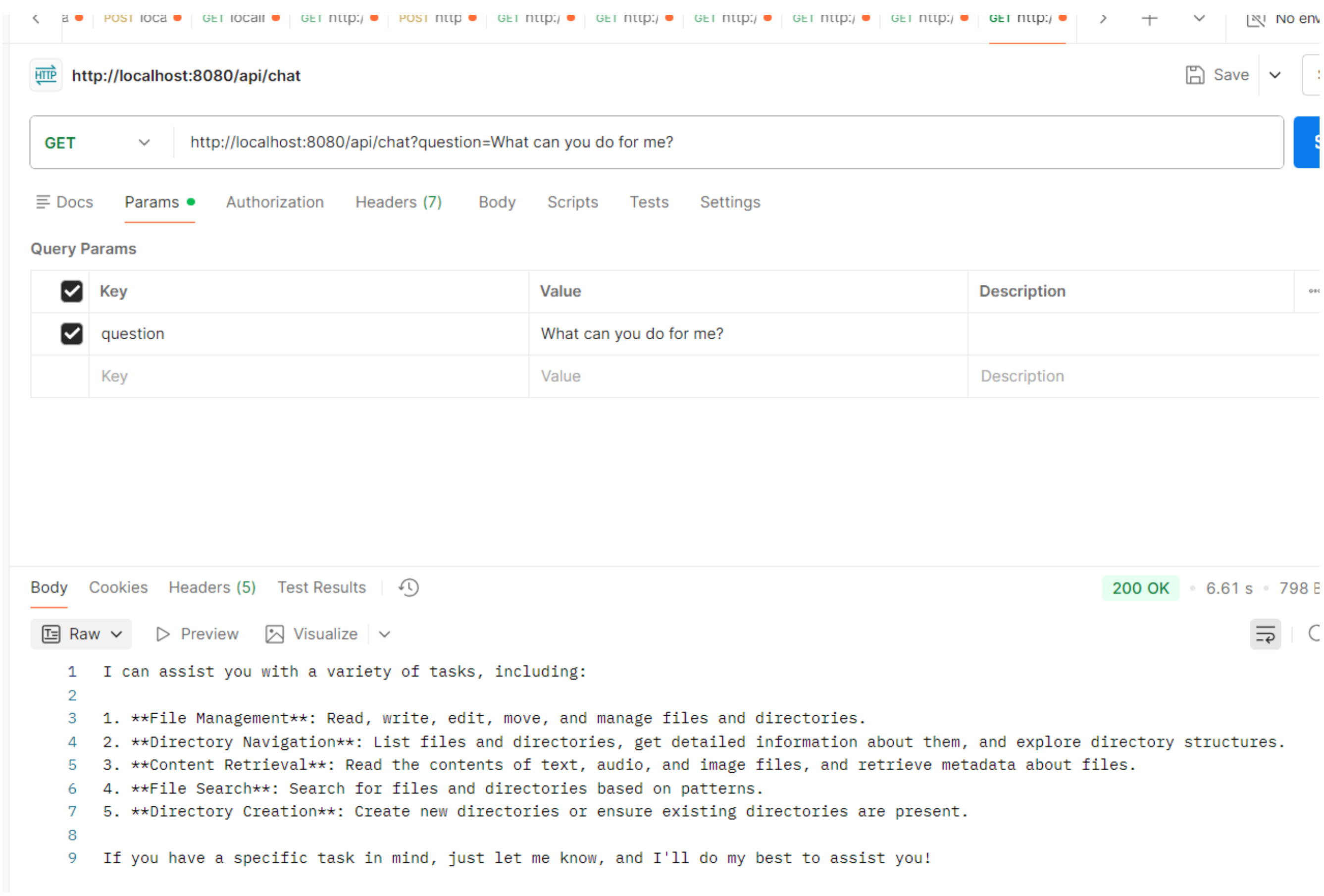
Ensure Node is in System Environment variables

This is for Windows add in the mcp-servers.json

```
      "command": "npx.cmd"
```

```
    public MCPController(ChatClient.Builder chatClientBuilder, ToolCallbackProvider
toolCallBackProvider){
        this.chatClient = chatClientBuilder
                .defaultToolCallbacks(toolCallBackProvider)
                .build();
    }
```

Now we ask the same question



GET http://localhost:8080/api/chat?question=What can you do for me?

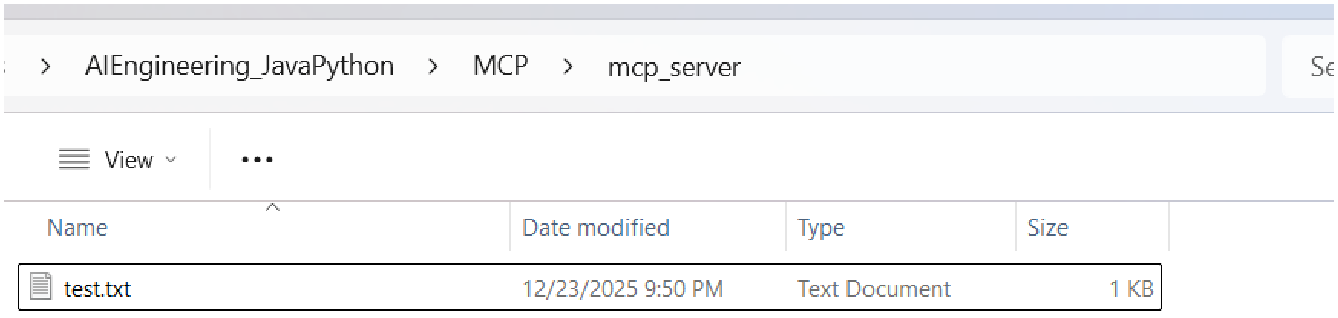
Query Params

Key	Value	Description
question	What can you do for me?	

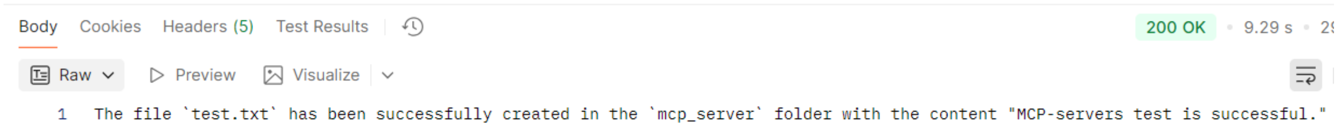
```
I can assist you with a variety of tasks, including:

1. **File Management**: Read, write, edit, move, and manage files and directories.
2. **Directory Navigation**: List files and directories, get detailed information about them, and explore directory structures.
3. **Content Retrieval**: Read the contents of text, audio, and image files, and retrieve metadata about files.
4. **File Search**: Search for files and directories based on patterns.
5. **Directory Creation**: Create new directories or ensure existing directories are present.

If you have a specific task in mind, just let me know, and I'll do my best to assist you!
```

GET http://localhost:8080/api/chat?question=What can you do for me?

It says File management, Directory navigation, Content retrieval

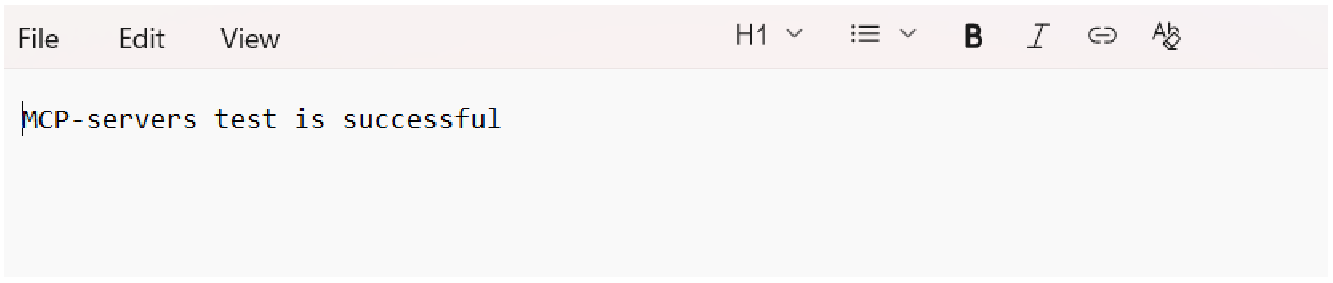


Name	Date modified	Type	Size
test.txt	12/23/2025 9:50 PM	Text Document	1 KB

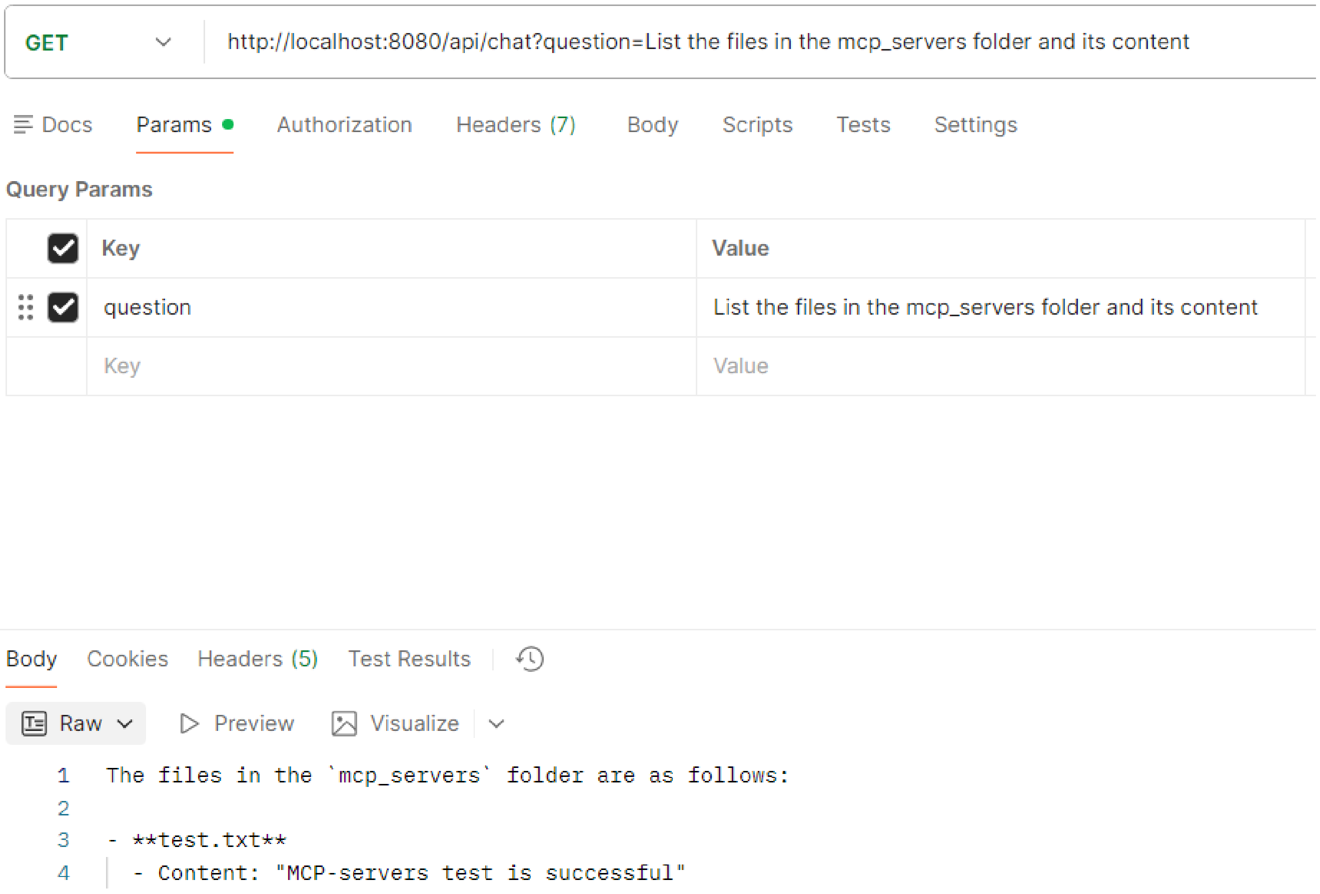


```
The file `test.txt` has been successfully created in the `mcp_server` folder with the content "MCP-servers test is successful."
```

File got created
The file `test.txt` has been successfully created in the `mcp_server` folder with the content "MCP-servers test is successful."



```
MCP-servers test is successful
```



```
GET http://localhost:8080/api/chat?question=List the files in the mcp_servers folder and its content
```

Query Params

Key	Value
question	List the files in the mcp_servers folder and its content
Key	Value

```
The files in the `mcp_servers` folder are as follows:

- **test.txt**
  - Content: "MCP-servers test is successful"
```