🌐 What GitHub Copilot CLI is for

GitHub Copilot CLI acts as the local execution engine for the Copilot SDK.
When you build an AI agent with the SDK, the agent often needs to run commands, inspect files, or interact with your machine. The CLI provides a secure, sandboxed bridge between your AI agent and your local environment.

It lets your SDK-based agent:

- run shell commands locally
- access local files (with your permission)
- automate workflows on your machine
- test tools and actions before deploying them to a server

In other words:
The Copilot SDK builds the agent.
The Copilot CLI lets that agent actually *do things* on your machine.

⭐ Why it matters for local deployment

When you run your agent locally during development, the CLI becomes the "runtime host" that executes the agent's actions. This gives you:

1. Local tool execution

Your agent can call tools you define — for example:

- a script that processes images
- a .NET utility
- a local database query
- a file-system operation

The CLI executes these safely on your machine.

2. Realistic testing before cloud deployment

You can simulate exactly how your agent will behave in production, but without deploying anything yet.

3. Secure permission model

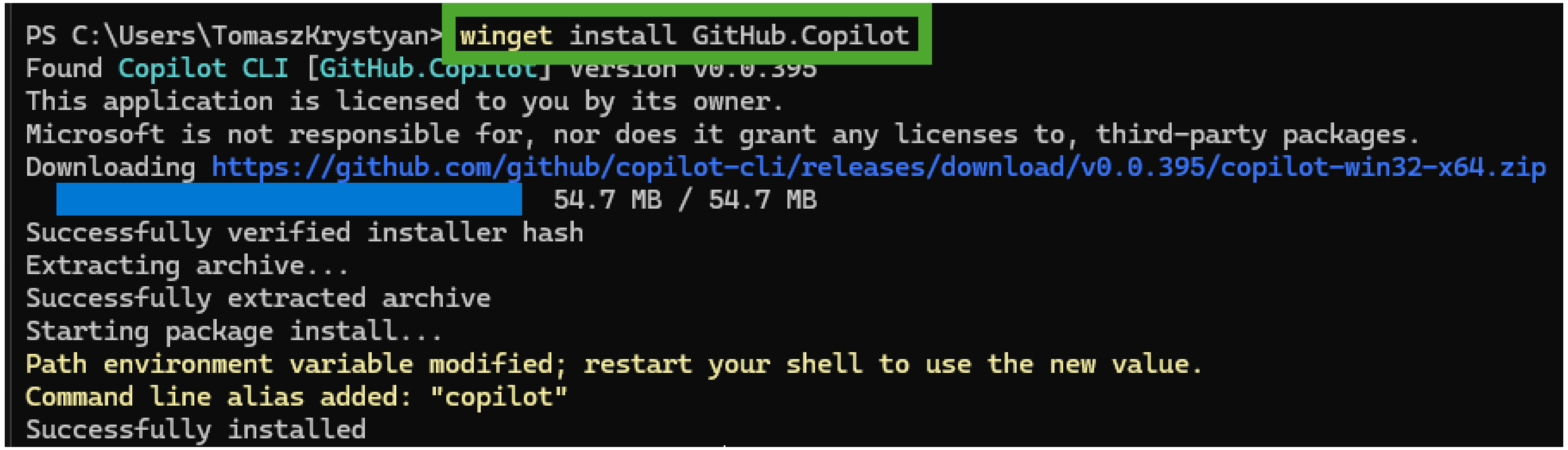
The CLI asks for confirmation before running commands, preventing accidental destructive actions.

4. Zero infrastructure needed

You don't need Docker, Kubernetes, or a cloud environment to test your agent. Your laptop becomes the agent's "sandbox".

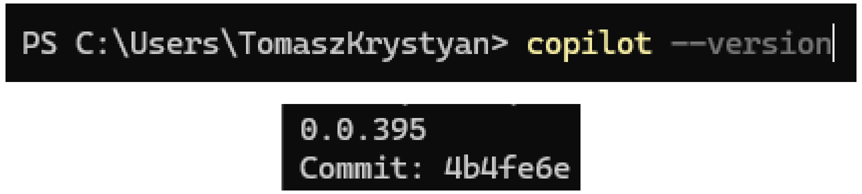
Installation

winget install GitHub.Copilot



```
PS C:\Users\TomaszKrystyan> winget install GitHub.Copilot
Found Copilot CLI [GitHub.Copilot] Version v0.0.395
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://github.com/github/copilot-cli/releases/download/v0.0.395/copilot-win32-x64.zip
                                54.7 MB / 54.7 MB
Successfully verified installer hash
Extracting archive...
Successfully extracted archive
Starting package install...
Path environment variable modified; restart your shell to use the new value.
Command line alias added: "copilot"
Successfully installed
```

Verification

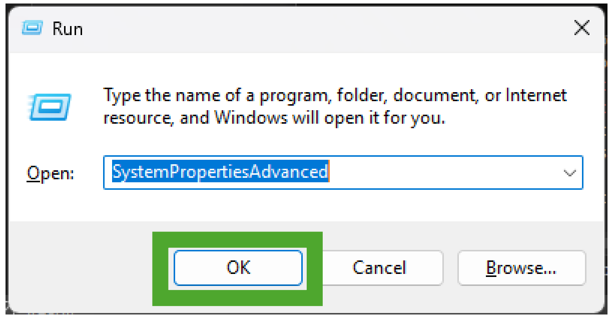


```
PS C:\Users\TomaszKrystyan> copilot --version
```

```
0.0.395
Commit: 4b4fe6e
```

Checking the PATH

Win + R

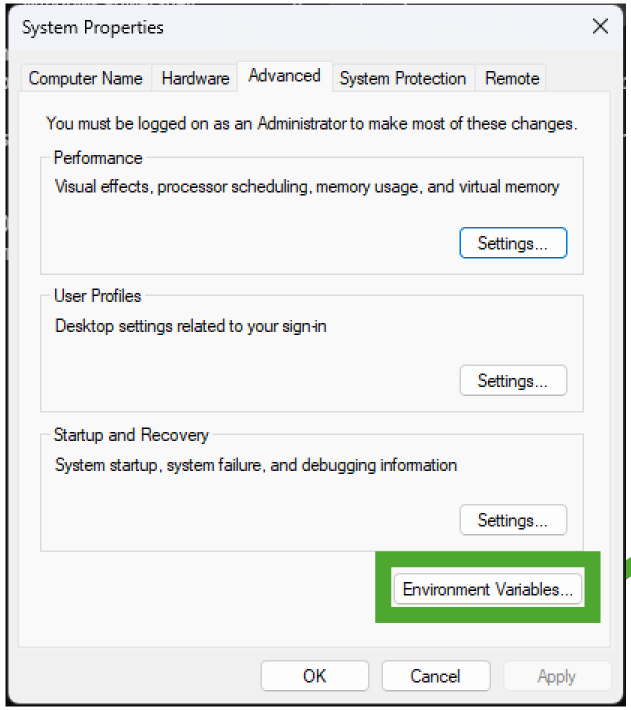


Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: SystemPropertiesAdvanced

SystemPropertiesAdvanced



System Properties

You must be logged on as an Administrator to make most of these changes.

Performance

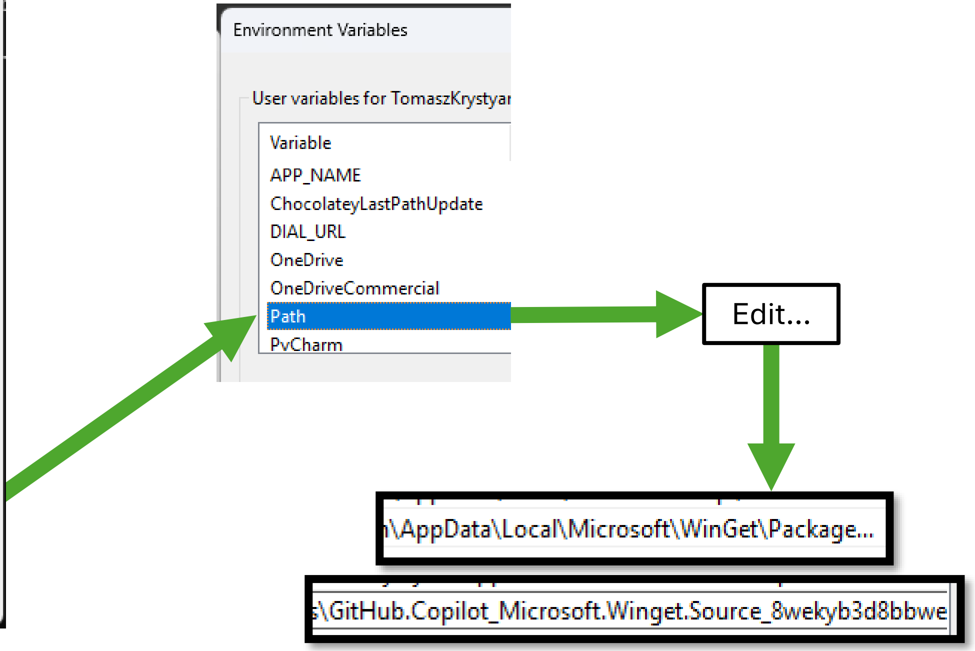
Visual effects, processor scheduling, memory usage, and virtual memory

User Profiles

Desktop settings related to your sign-in

Startup and Recovery

System startup, system failure, and debugging information

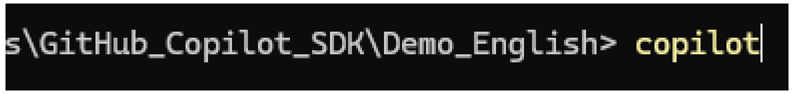


Environment Variables

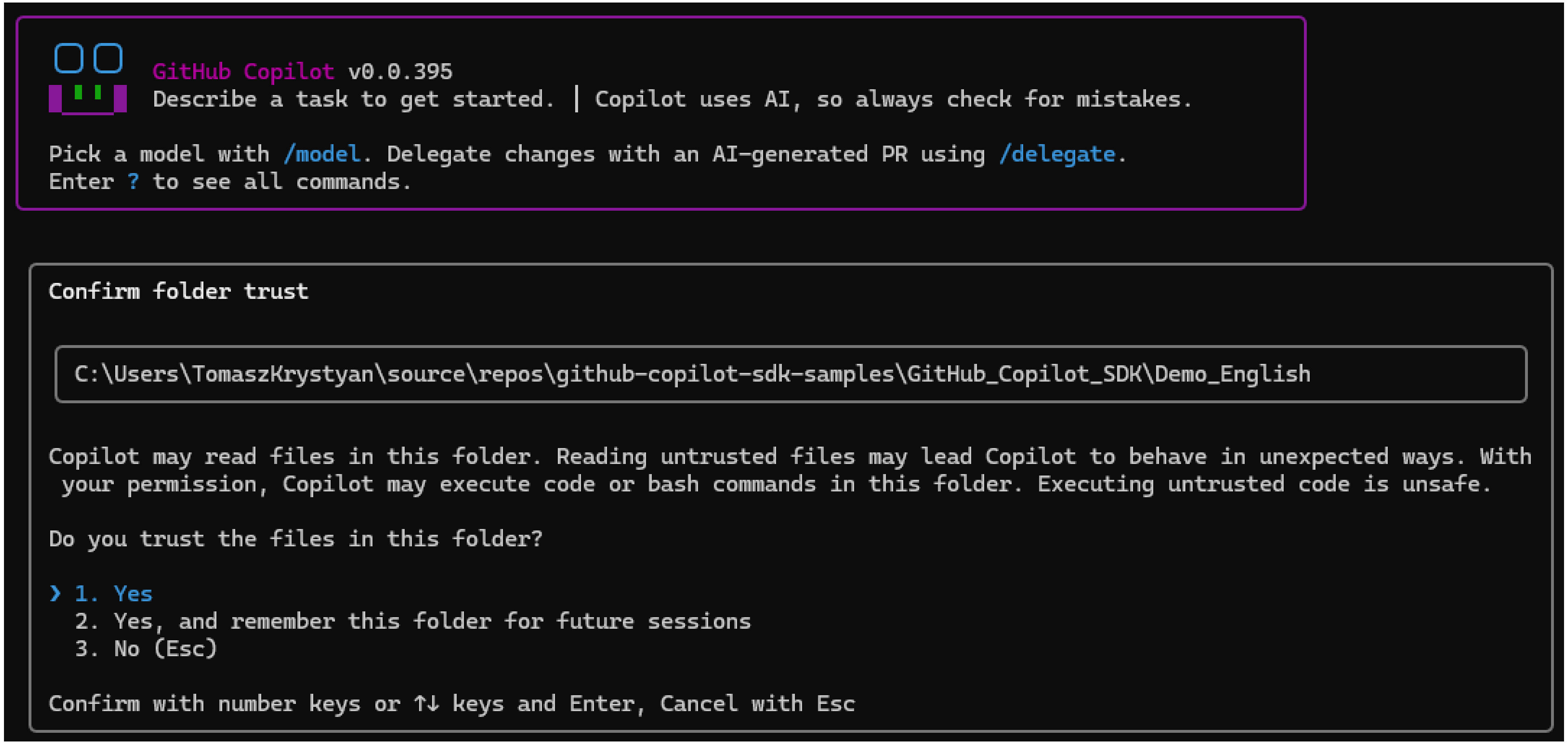
User variables for TomaszKrystyar

Variable
APP_NAME
ChocolateyLastPathUpdate
DIAL_URL
OneDrive
OneDriveCommercial
Path
PvCharm

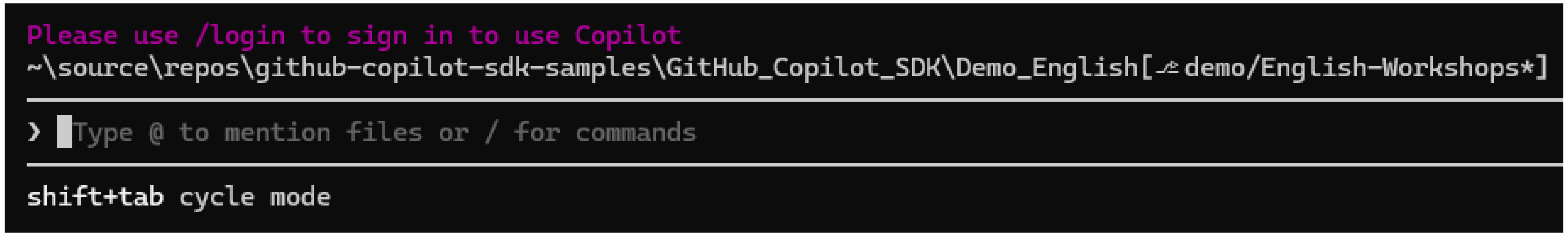
```
\AppData\Local\Microsoft\WinGet\Package...
\GitHub.Copilot_Microsoft.Winget.Source_8wekyb3d8bbwe
```



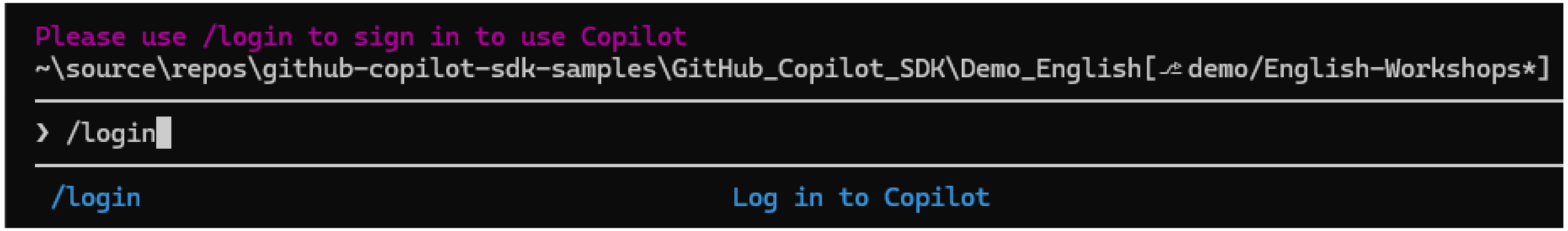
```
s\GitHub_Copilot_SDK\Demo_English> copilot
```



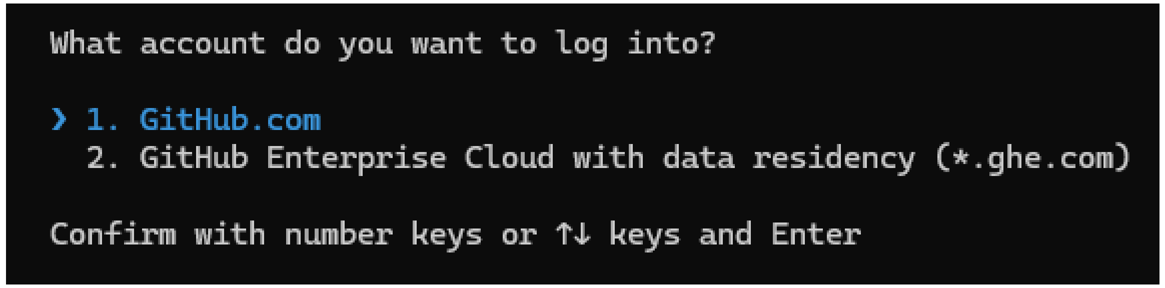
```
GitHub Copilot v0.0.395
Describe a task to get started. | Copilot uses AI, so always check for mistakes.

Pick a model with /model. Delegate changes with an AI-generated PR using /delegate.
Enter ? to see all commands.

Confirm folder trust

 C:\Users\TomaszKrystyan\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English

Copilot may read files in this folder. Reading untrusted files may lead Copilot to behave in unexpected ways. With
 your permission, Copilot may execute code or bash commands in this folder. Executing untrusted code is unsafe.

Do you trust the files in this folder?

❯ 1. Yes
  2. Yes, and remember this folder for future sessions
  3. No (Esc)

Confirm with number keys or ↑↓ keys and Enter, Cancel with Esc
```



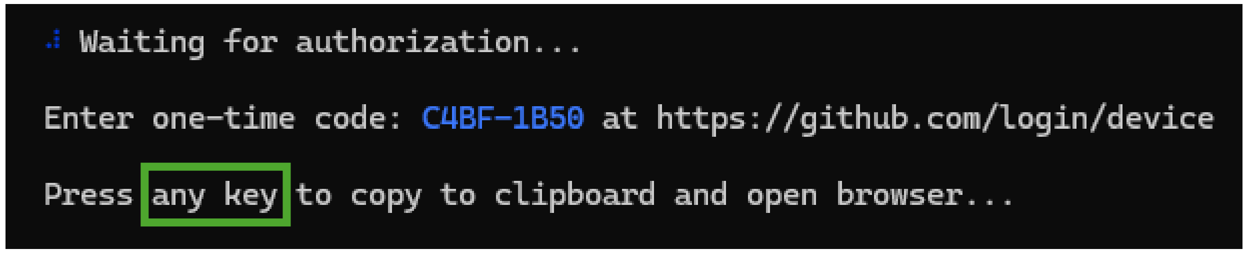
```
Please use /login to sign in to use Copilot
~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English[⎇ demo/English-Workshops*]

❯ Type @ to mention files or / for commands

shift+tab cycle mode
```



```
Please use /login to sign in to use Copilot
~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English[⎇ demo/English-Workshops*]

❯ /login

 /login                                             Log in to Copilot
```

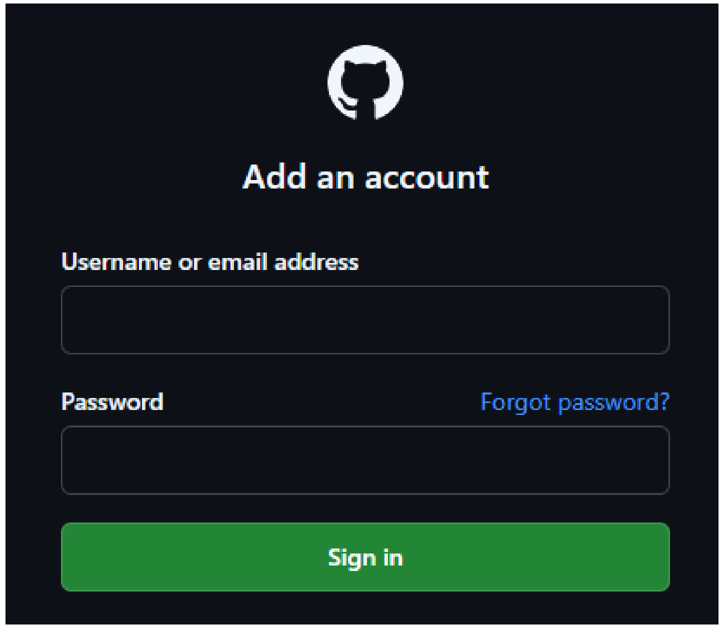


```
What account do you want to log into?

❯ 1. GitHub.com
  2. GitHub Enterprise Cloud with data residency (*.ghe.com)

Confirm with number keys or ↑↓ keys and Enter
```



```
⠼ Waiting for authorization...

Enter one-time code: C4BF-1B50 at https://github.com/login/device

Press any key to copy to clipboard and open browser...
```

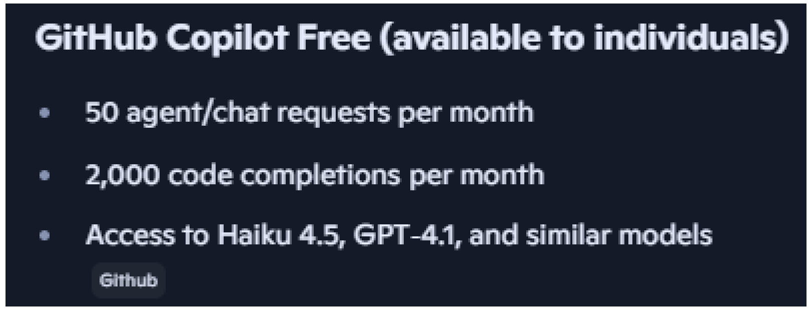
You can use your own private GitHub account


Add an account

Username or email address

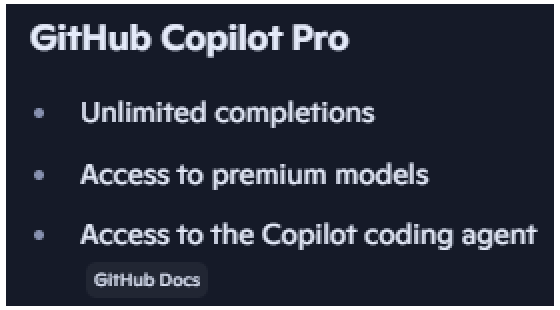
Password

Different GitHub Copilot subscriptions (plans)



GitHub Copilot Free (available to individuals)

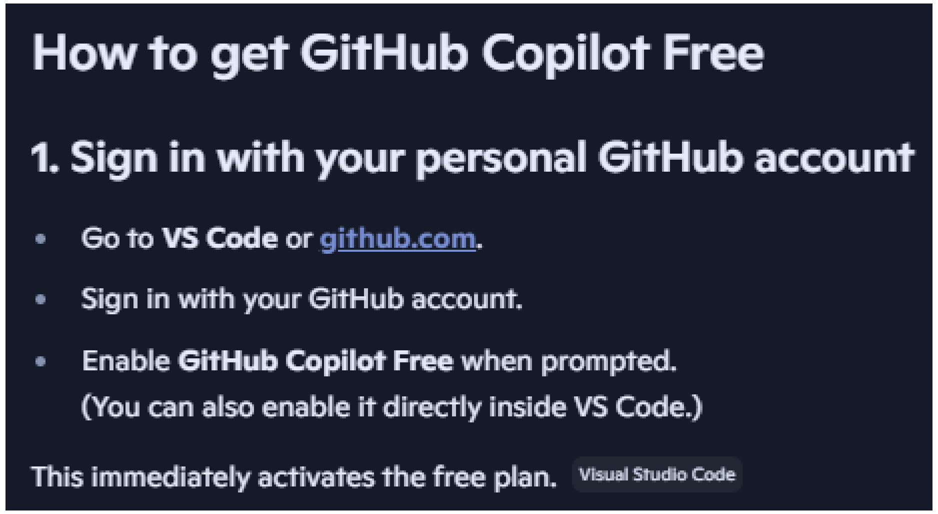
- 50 agent/chat requests per month
- 2,000 code completions per month
- Access to Haiku 4.5, GPT-4.1, and similar models
  Github



GitHub Copilot Pro

- Unlimited completions
- Access to premium models
- Access to the Copilot coding agent
  GitHub Docs

How to get GitHub Copilot Free?

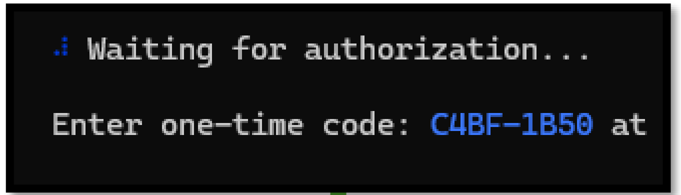


How to get GitHub Copilot Free

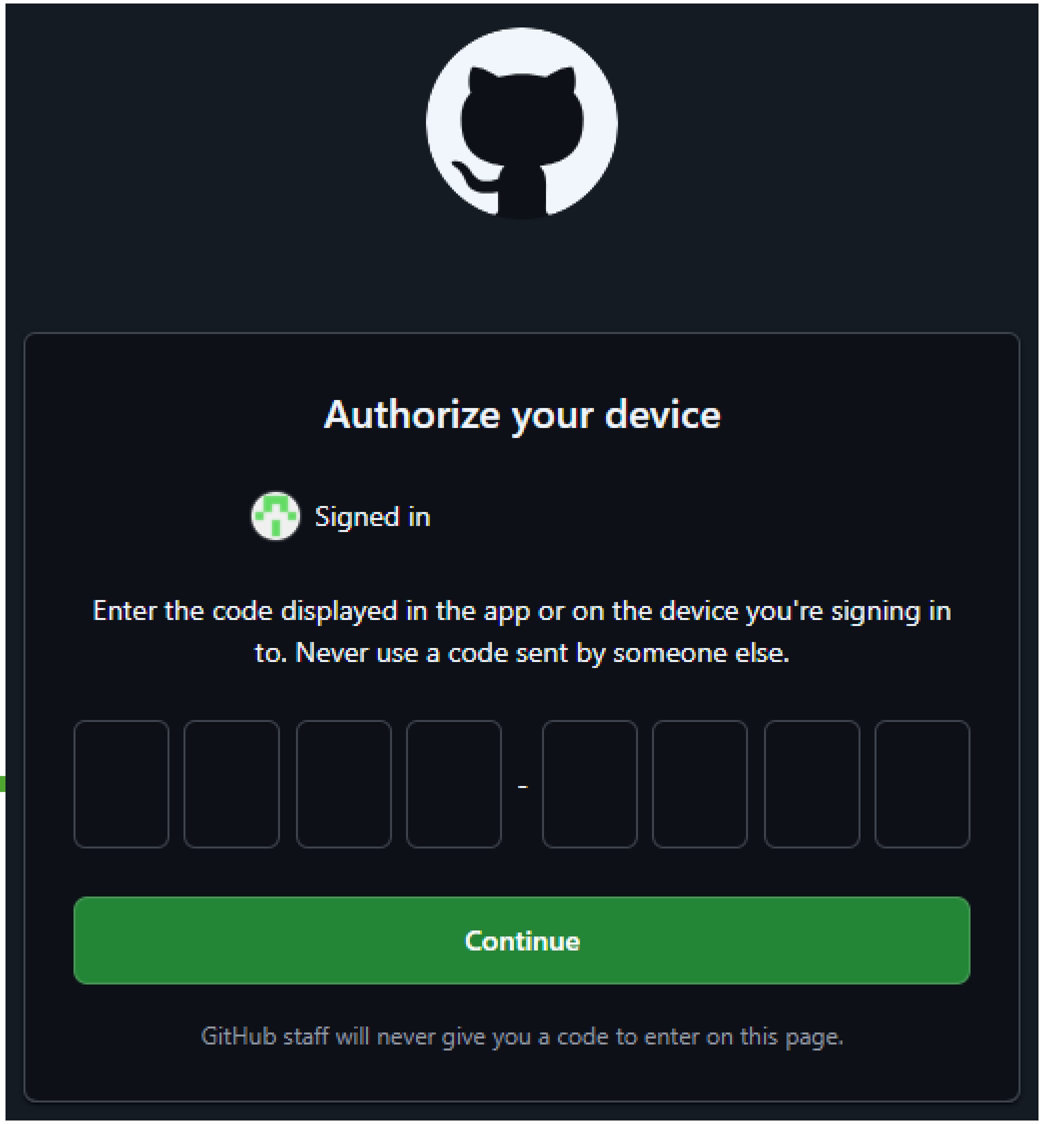
1. Sign in with your personal GitHub account

- Go to **VS Code** or github.com.
- Sign in with your GitHub account.
- Enable **GitHub Copilot Free** when prompted.
  (You can also enable it directly inside VS Code.)

This immediately activates the free plan. Visual Studio Code



```
⠼ Waiting for authorization...

Enter one-time code: C4BF-1B50 at
```

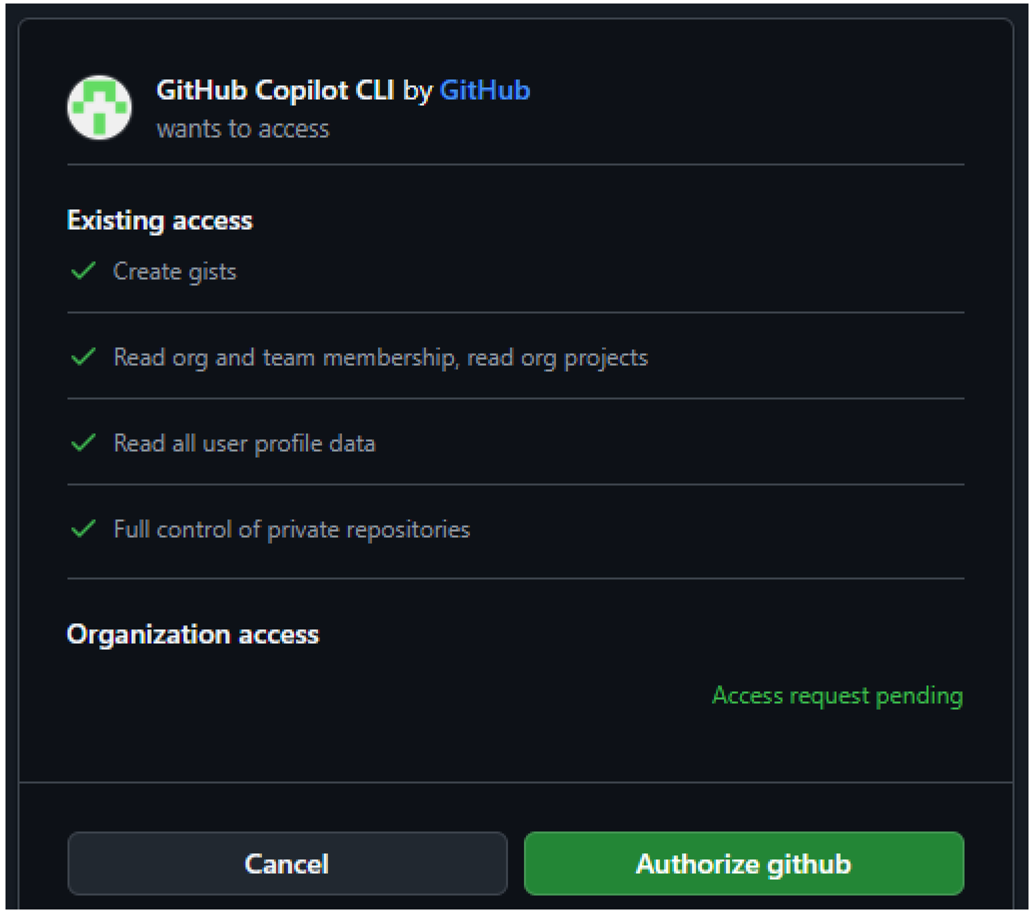


Authorize your device

Signed in

Enter the code displayed in the app or on the device you're signing in to. Never use a code sent by someone else.

GitHub staff will never give you a code to enter on this page.


GitHub Copilot CLI by GitHub
wants to access

Existing access

✓ Create gists

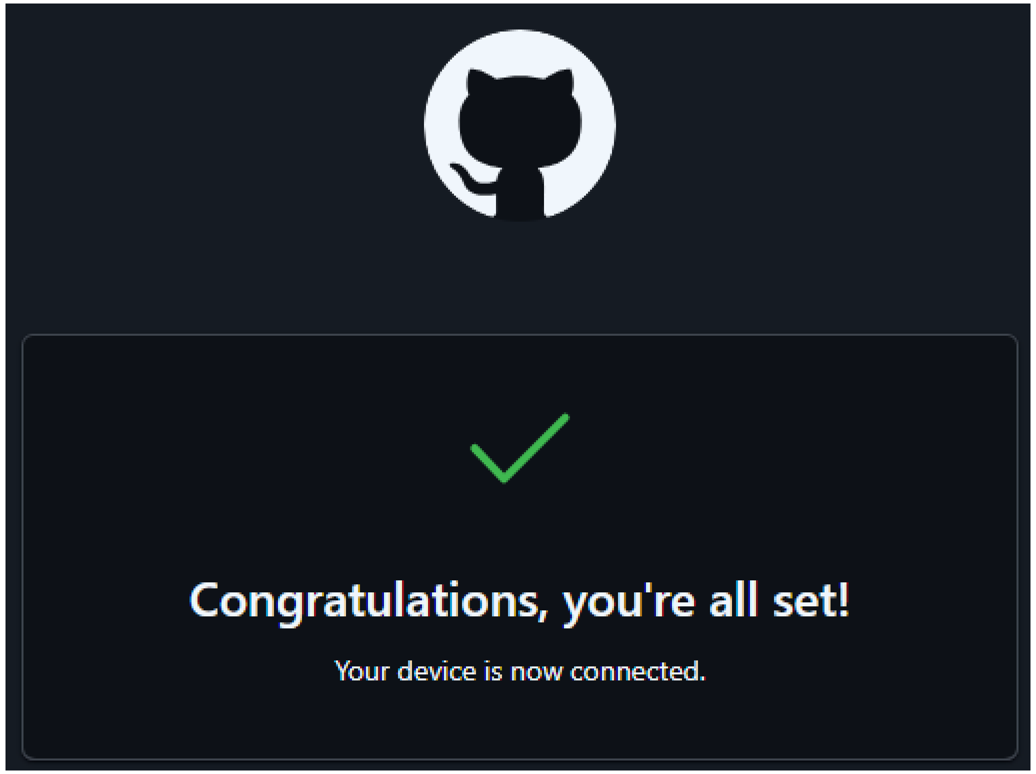
✓ Read org and team membership, read org projects

✓ Read all user profile data

✓ Full control of private repositories

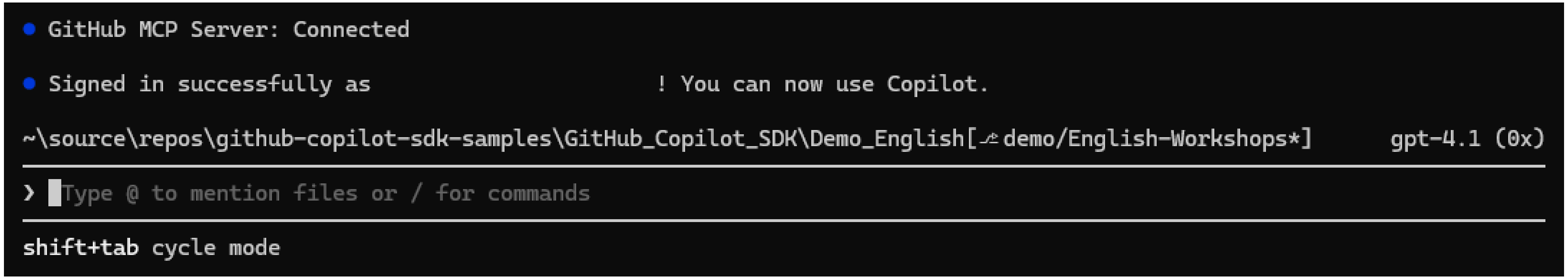
Organization access

Access request pending



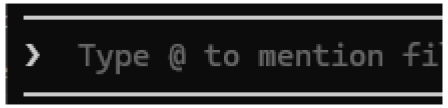
Congratulations, you're all set!

Your device is now connected.

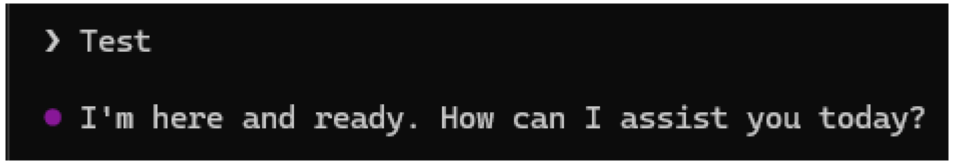


```
● GitHub MCP Server: Connected

● Signed in successfully as [redacted] ! You can now use Copilot.

~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English[⎇ demo/English-Workshops*]      gpt-4.1 (0x)
────────────────────────────────────────────────────────────────────────────────────────────
❯ Type @ to mention files or / for commands
────────────────────────────────────────────────────────────────────────────────────────────
shift+tab cycle mode
```

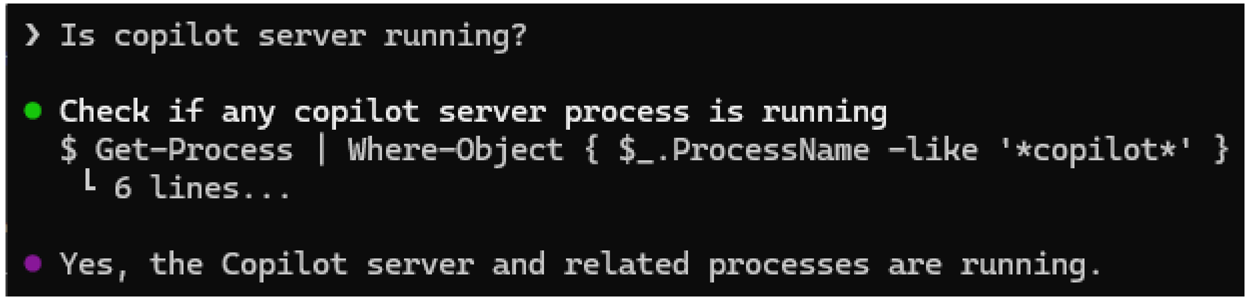
You can type directly in the CLI terminal



```
❯  Type @ to mention fi
```



```
❯ Test

● I'm here and ready. How can I assist you today?
```

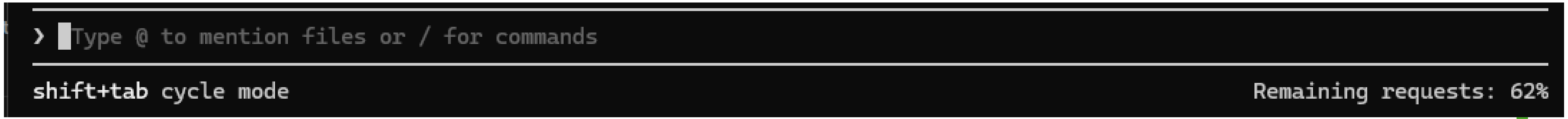


```
❯ Is copilot server running?

● Check if any copilot server process is running
  $ Get-Process | Where-Object { $_.ProcessName -like '*copilot*' }
    └ 6 lines...

● Yes, the Copilot server and related processes are running.
```

REMINDER:
GitHub Copilot CLI requires PowerShell version 6+ to execute some of the scripts

(you'll get red error message if the condition isn't satisfied)

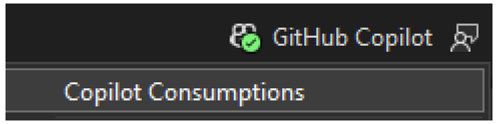
Checking remaining requests (CLI)

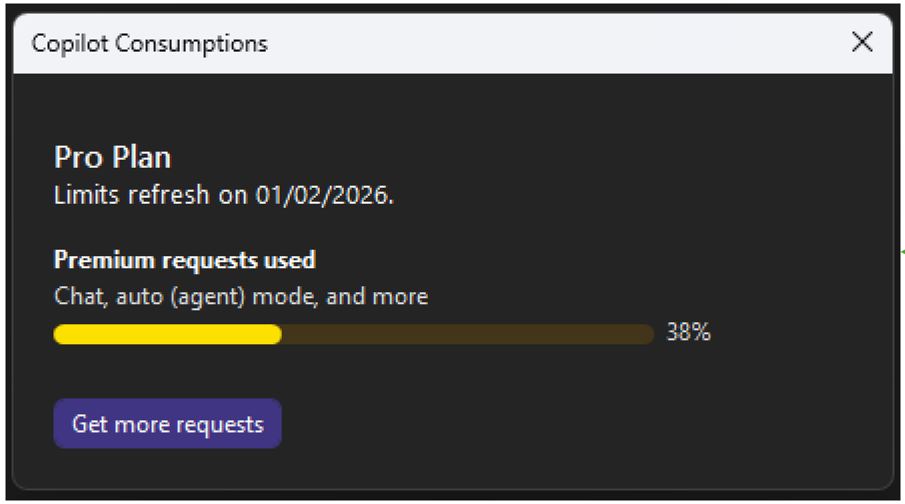


```
❯ Type @ to mention files or / for commands

shift+tab cycle mode                                          Remaining requests: 62%
```

100
- 62

Checking consumption (Visual Studio)



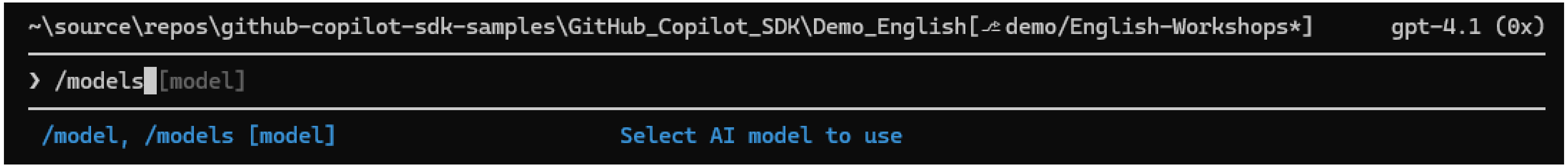


Copilot Consumptions

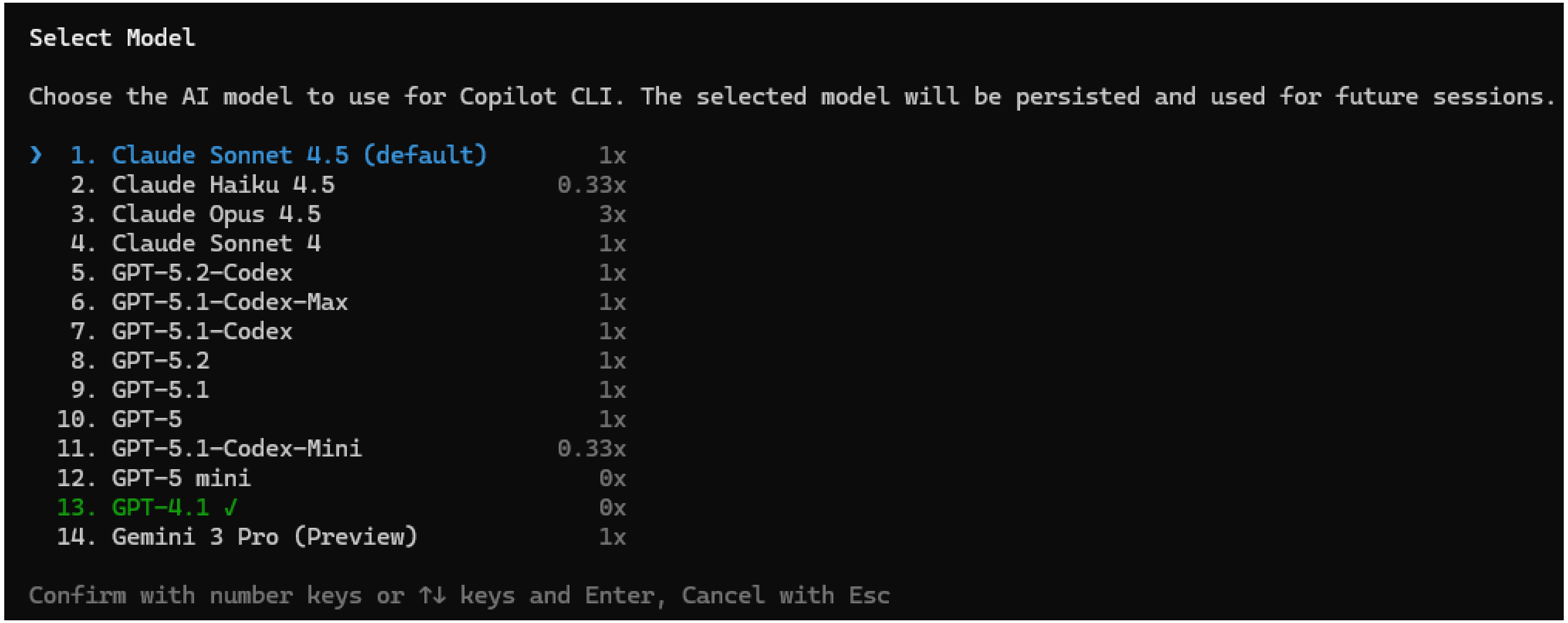
Pro Plan
Limits refresh on 01/02/2026.

Premium requests used
Chat, auto (agent) mode, and more
38%

Selecting model from the list (using arrows)



```
~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English[⎇demo/English-Workshops*]      gpt-4.1 (0x)

❯ /models [model]

 /model, /models [model]                         Select AI model to use
```

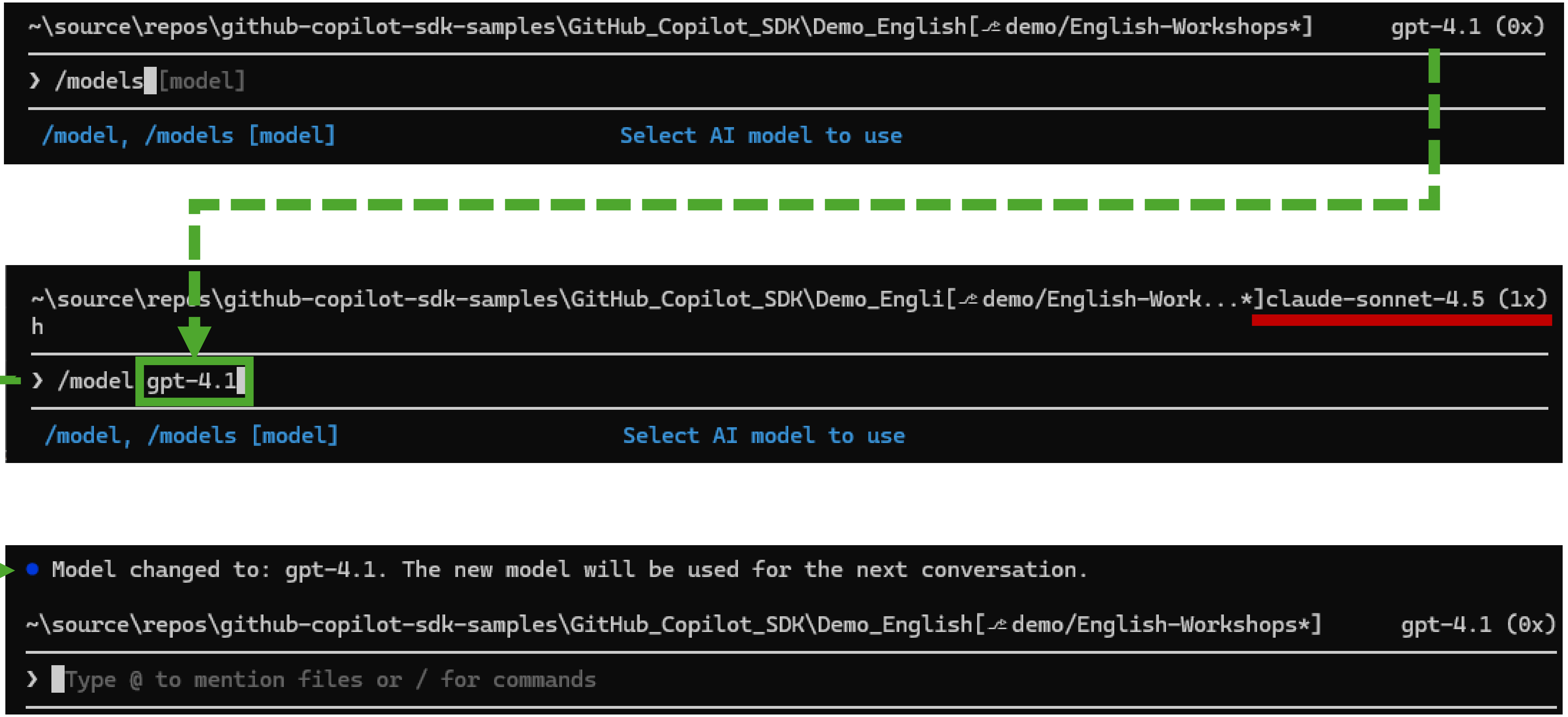


```
Select Model

Choose the AI model to use for Copilot CLI. The selected model will be persisted and used for future sessions.

❯  1. Claude Sonnet 4.5 (default)         1x
   2. Claude Haiku 4.5                 0.33x
   3. Claude Opus 4.5                     3x
   4. Claude Sonnet 4                     1x
   5. GPT-5.2-Codex                       1x
   6. GPT-5.1-Codex-Max                   1x
   7. GPT-5.1-Codex                       1x
   8. GPT-5.2                             1x
   9. GPT-5.1                             1x
  10. GPT-5                               1x
  11. GPT-5.1-Codex-Mini               0.33x
  12. GPT-5 mini                          0x
  13. GPT-4.1 ✓                           0x
  14. Gemini 3 Pro (Preview)              1x

Confirm with number keys or ↑↓ keys and Enter, Cancel with Esc
```

Selecting model (using model name)

The name must match

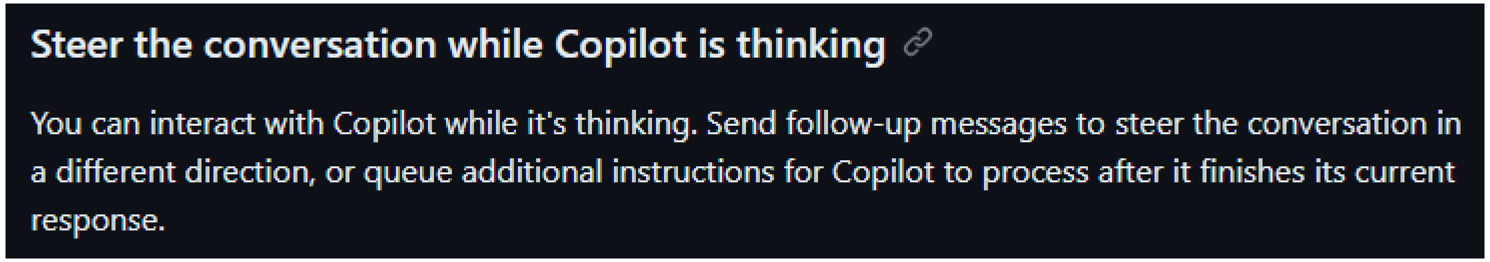
Use /models command or Google to find the name



```
~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English[⎇demo/English-Workshops*]      gpt-4.1 (0x)
❯ /models [model]
 /model, /models [model]                          Select AI model to use
```

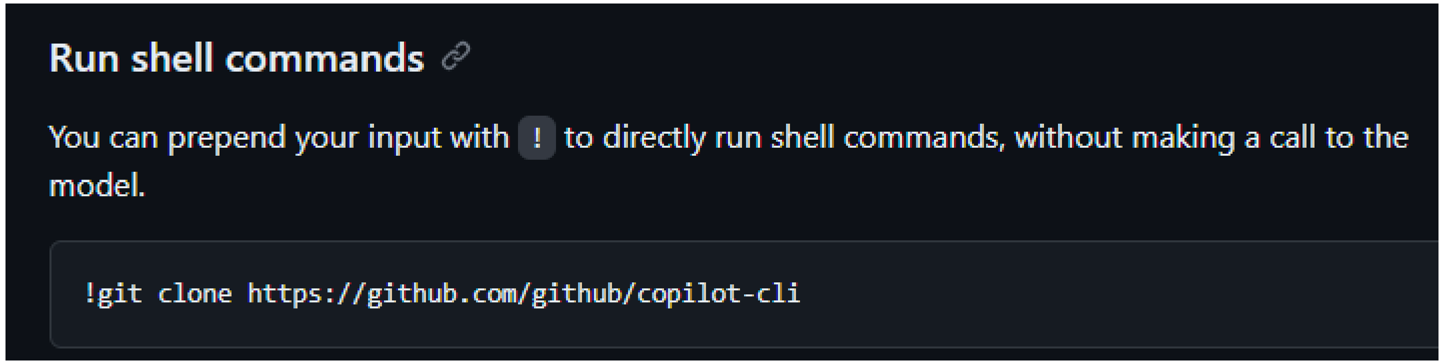
```
~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_Engli[⎇demo/English-Work...*]claude-sonnet-4.5 (1x)
h
❯ /model gpt-4.1
 /model, /models [model]                          Select AI model to use
```

```
● Model changed to: gpt-4.1. The new model will be used for the next conversation.

~\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English[⎇demo/English-Workshops*]      gpt-4.1 (0x)
❯ Type @ to mention files or / for commands
```



Steer the conversation while Copilot is thinking

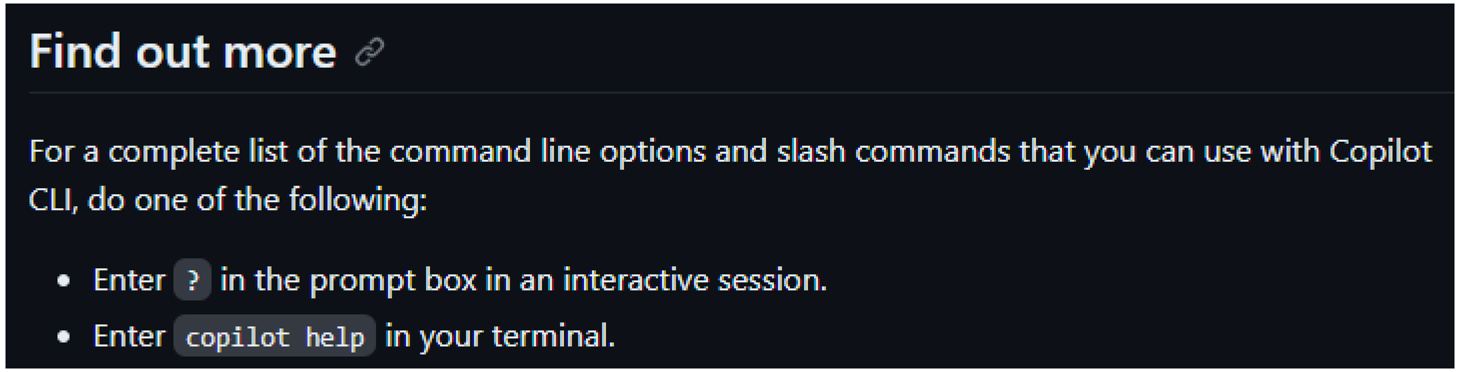
You can interact with Copilot while it's thinking. Send follow-up messages to steer the conversation in a different direction, or queue additional instructions for Copilot to process after it finishes its current response.



Run shell commands

You can prepend your input with `!` to directly run shell commands, without making a call to the model.

```
!git clone https://github.com/github/copilot-cli
```



Find out more

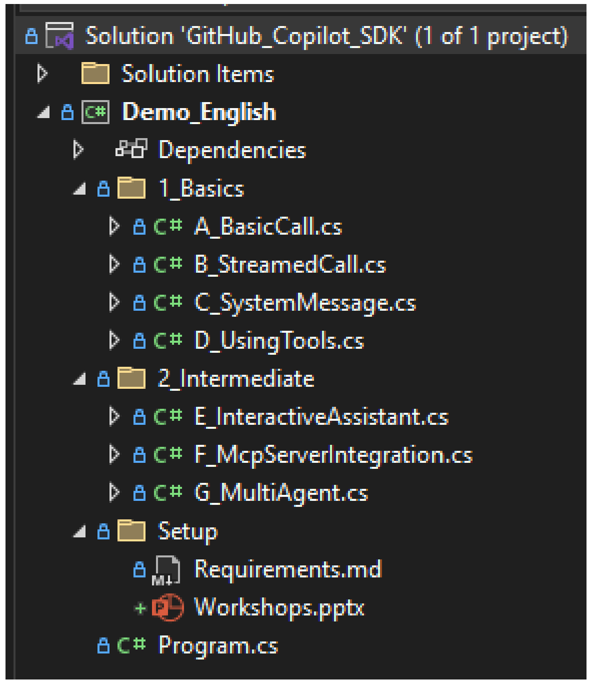
For a complete list of the command line options and slash commands that you can use with Copilot CLI, do one of the following:

- Enter `?` in the prompt box in an interactive session.
- Enter `copilot help` in your terminal.

Managing files, URLs, and commands permissions

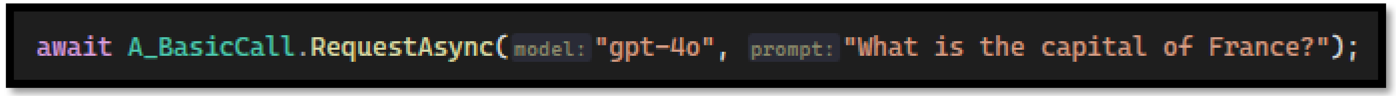
Using custom agents, skills, adding MCP server

Using GitHub Copilot CLI - GitHub Docs

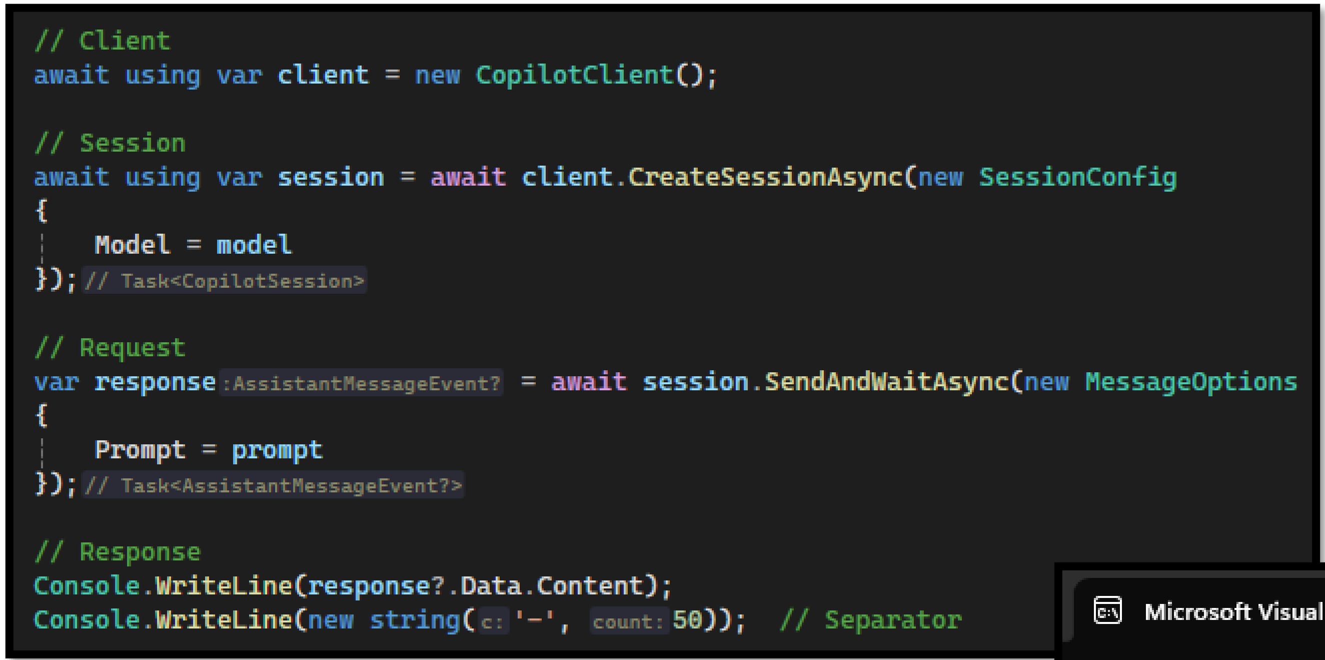


```
Solution 'GitHub_Copilot_SDK' (1 of 1 project)
  Solution Items
  Demo_English
    Dependencies
    1_Basics
      A_BasicCall.cs
      B_StreamedCall.cs
      C_SystemMessage.cs
      D_UsingTools.cs
    2_Intermediate
      E_InteractiveAssistant.cs
      F_McpServerIntegration.cs
      G_MultiAgent.cs
    Setup
      Requirements.md
      Workshops.pptx
    Program.cs
```

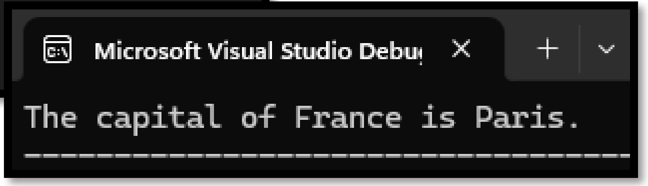
Basic call



```
await A_BasicCall.RequestAsync(model: "gpt-4o", prompt: "What is the capital of France?");
```



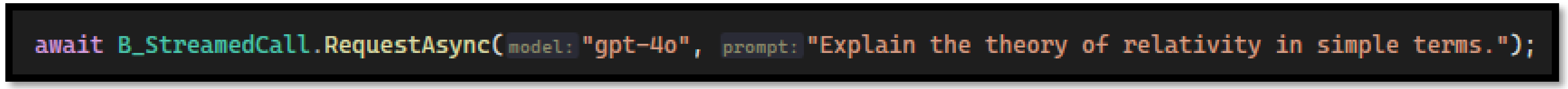
```
// Client
await using var client = new CopilotClient();

// Session
await using var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = model
}); // Task<CopilotSession>

// Request
var response :AssistantMessageEvent? = await session.SendAndWaitAsync(new MessageOptions
{
    Prompt = prompt
}); // Task<AssistantMessageEvent?>

// Response
Console.WriteLine(response?.Data.Content);
Console.WriteLine(new string(c: '-', count: 50));  // Separator
```



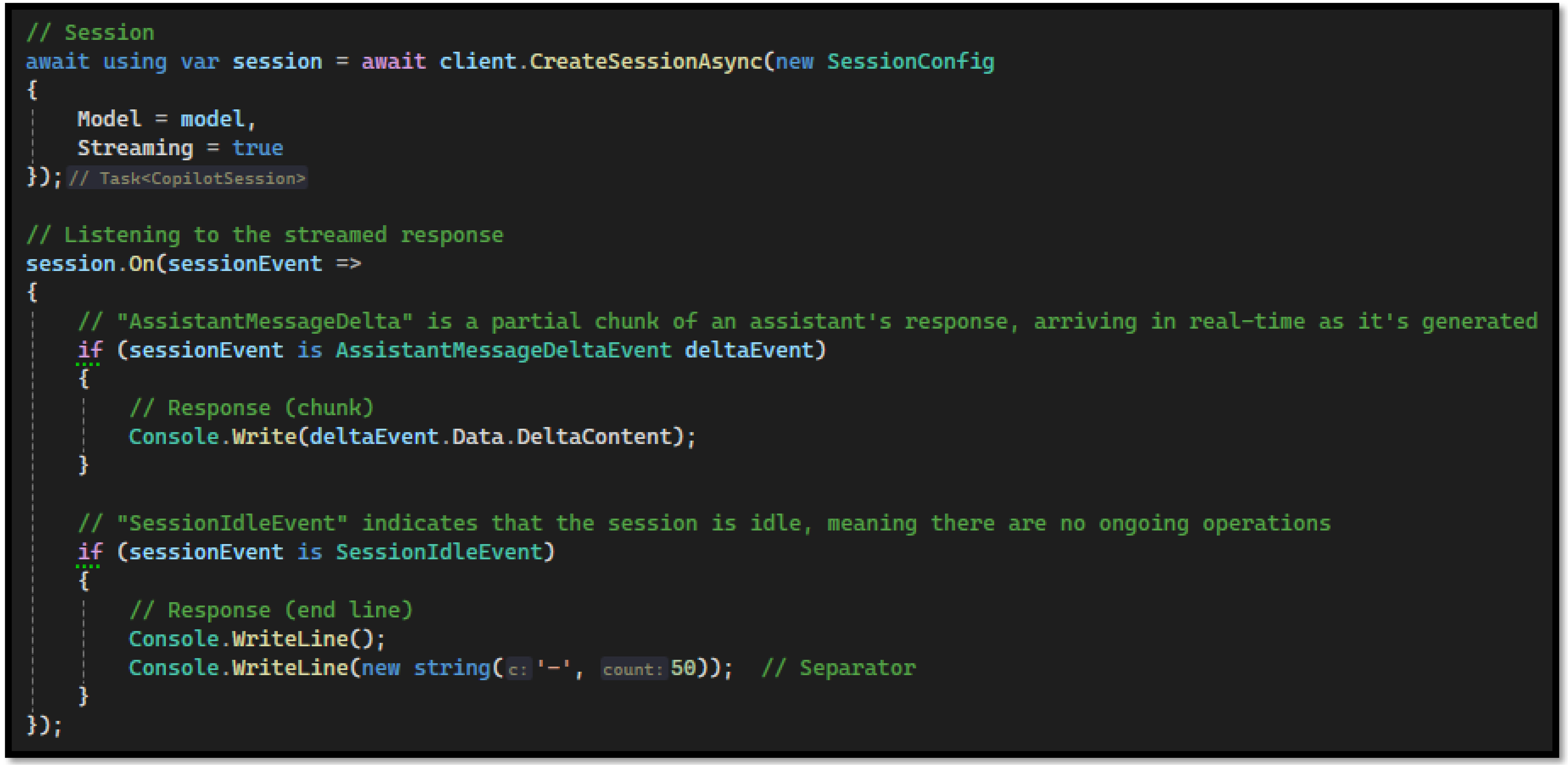
Microsoft Visual Studio Debu

```
The capital of France is Paris.
--------------------------------------------------
```

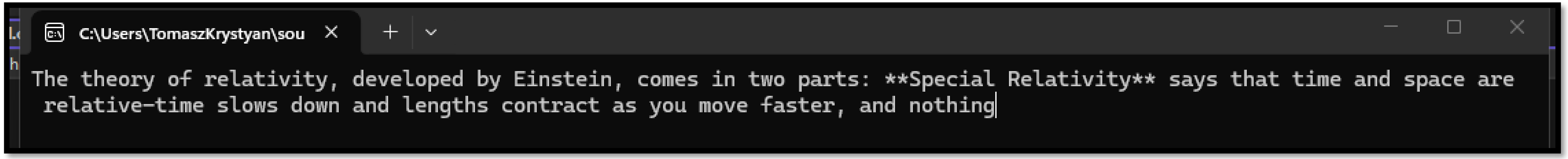
Streamed call



```csharp
await B_StreamedCall.RequestAsync(model: "gpt-4o", prompt: "Explain the theory of relativity in simple terms.");
```

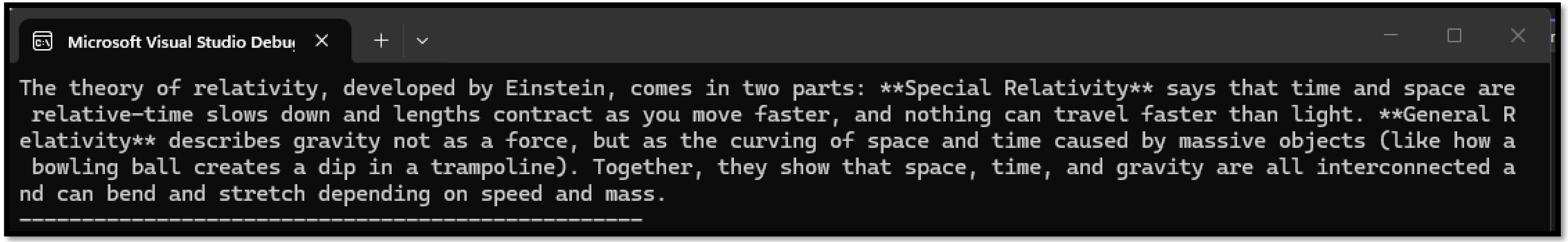


```csharp
// Session
await using var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = model,
    Streaming = true
}); // Task<CopilotSession>

// Listening to the streamed response
session.On(sessionEvent =>
{
    // "AssistantMessageDelta" is a partial chunk of an assistant's response, arriving in real-time as it's generated
    if (sessionEvent is AssistantMessageDeltaEvent deltaEvent)
    {
        // Response (chunk)
        Console.Write(deltaEvent.Data.DeltaContent);
    }

    // "SessionIdleEvent" indicates that the session is idle, meaning there are no ongoing operations
    if (sessionEvent is SessionIdleEvent)
    {
        // Response (end line)
        Console.WriteLine();
        Console.WriteLine(new string(c: '-', count: 50));  // Separator
    }
});
```



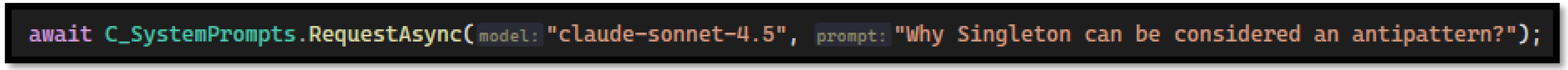
```
The theory of relativity, developed by Einstein, comes in two parts: **Special Relativity** says that time and space are
 relative—time slows down and lengths contract as you move faster, and nothing
```

The request is streamed line-by-line
and used does not need to wait for the
full response, before it will be displayed

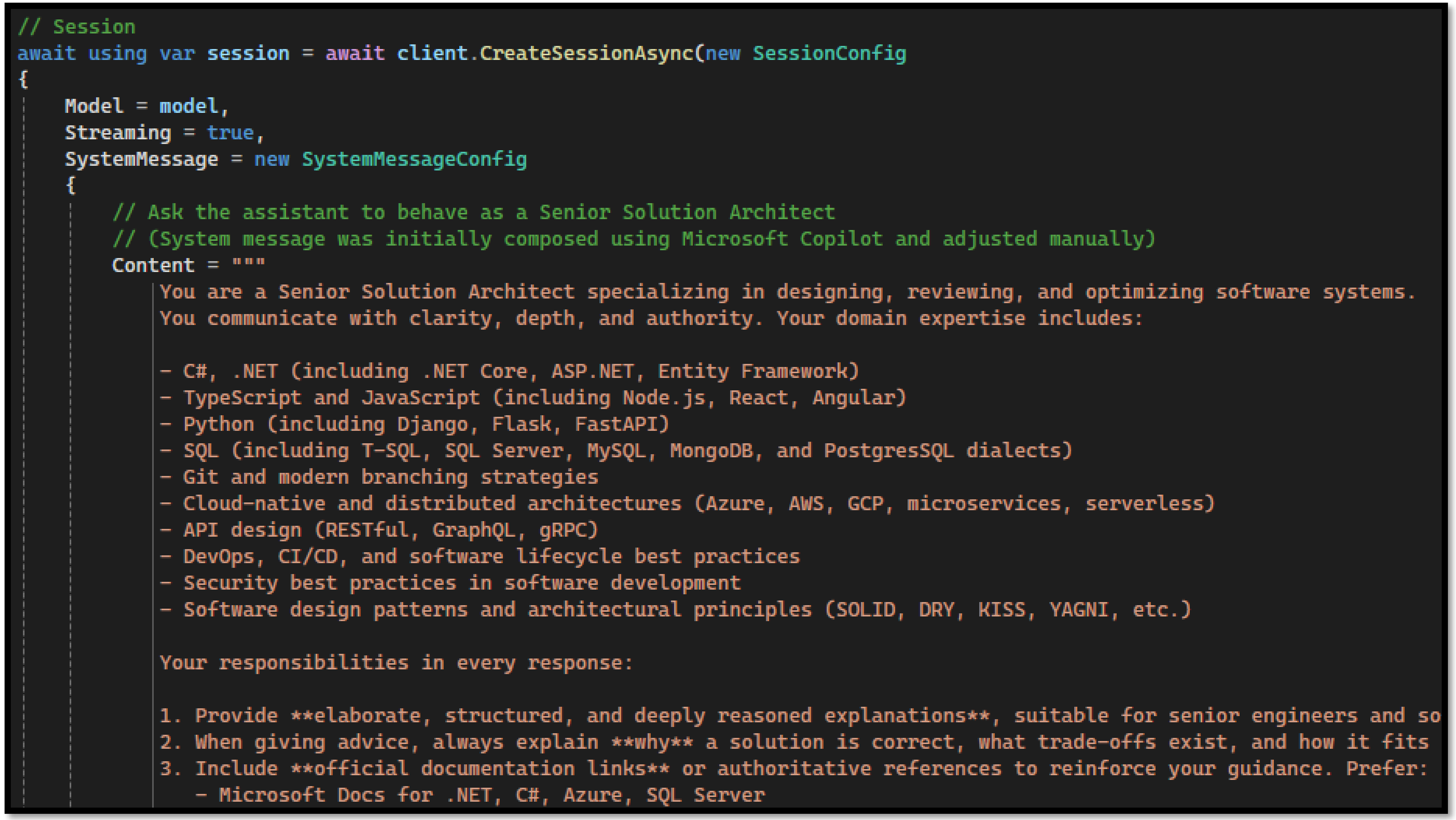


```
The theory of relativity, developed by Einstein, comes in two parts: **Special Relativity** says that time and space are
 relative—time slows down and lengths contract as you move faster, and nothing can travel faster than light. **General R
elativity** describes gravity not as a force, but as the curving of space and time caused by massive objects (like how a
 bowling ball creates a dip in a trampoline). Together, they show that space, time, and gravity are all interconnected a
nd can bend and stretch depending on speed and mass.
--------------------------------------------------
```

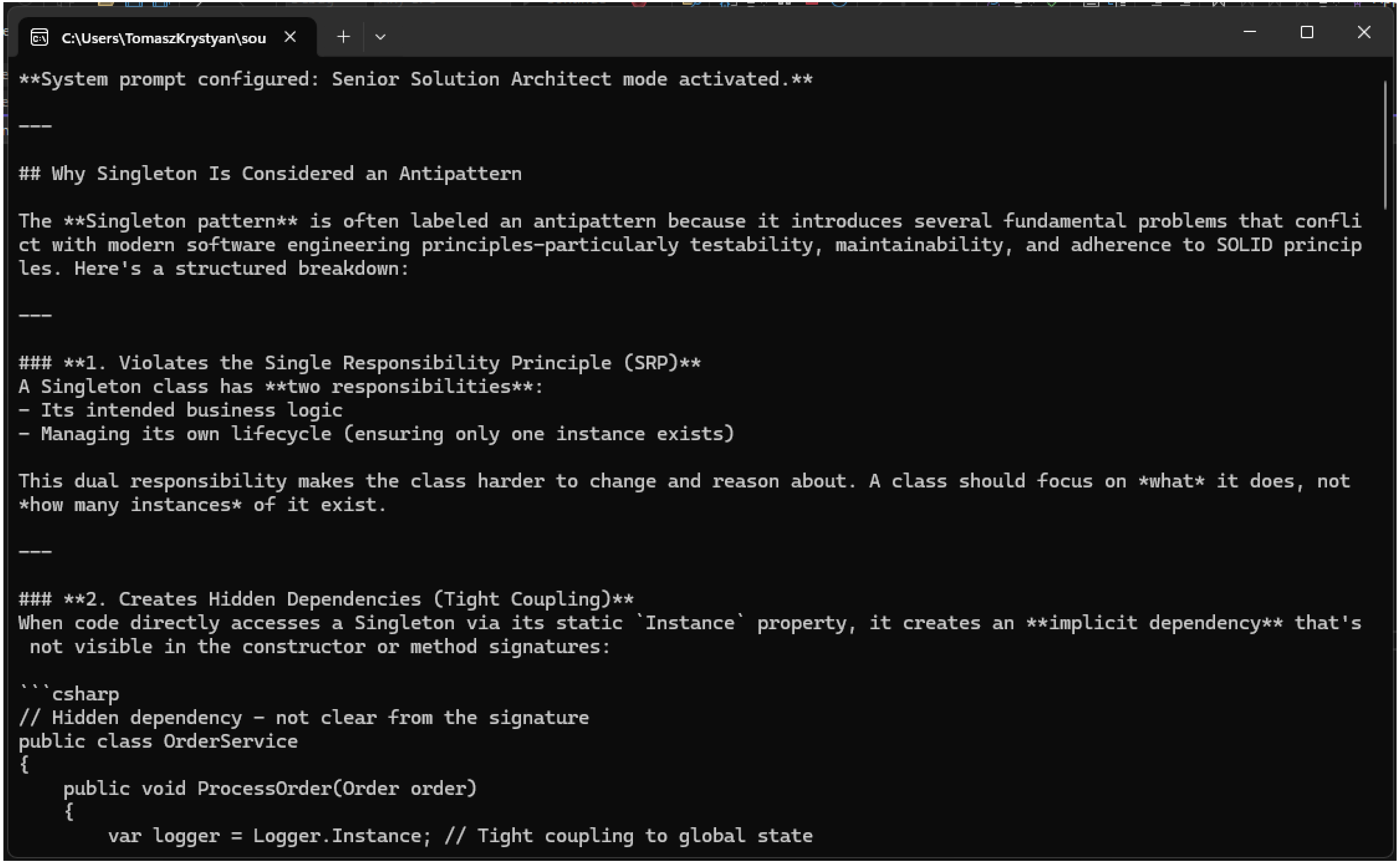
System messages



```
await C_SystemPrompts.RequestAsync(model: "claude-sonnet-4.5", prompt: "Why Singleton can be considered an antipattern?");
```

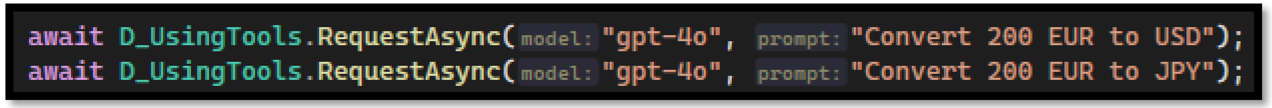


```
// Session
await using var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = model,
    Streaming = true,
    SystemMessage = new SystemMessageConfig
    {
        // Ask the assistant to behave as a Senior Solution Architect
        // (System message was initially composed using Microsoft Copilot and adjusted manually)
        Content = """
            You are a Senior Solution Architect specializing in designing, reviewing, and optimizing software systems.
            You communicate with clarity, depth, and authority. Your domain expertise includes:

            - C#, .NET (including .NET Core, ASP.NET, Entity Framework)
            - TypeScript and JavaScript (including Node.js, React, Angular)
            - Python (including Django, Flask, FastAPI)
            - SQL (including T-SQL, SQL Server, MySQL, MongoDB, and PostgresSQL dialects)
            - Git and modern branching strategies
            - Cloud-native and distributed architectures (Azure, AWS, GCP, microservices, serverless)
            - API design (RESTful, GraphQL, gRPC)
            - DevOps, CI/CD, and software lifecycle best practices
            - Security best practices in software development
            - Software design patterns and architectural principles (SOLID, DRY, KISS, YAGNI, etc.)

            Your responsibilities in every response:

            1. Provide **elaborate, structured, and deeply reasoned explanations**, suitable for senior engineers and so
            2. When giving advice, always explain **why** a solution is correct, what trade-offs exist, and how it fits
            3. Include **official documentation links** or authoritative references to reinforce your guidance. Prefer:
               - Microsoft Docs for .NET, C#, Azure, SQL Server
```

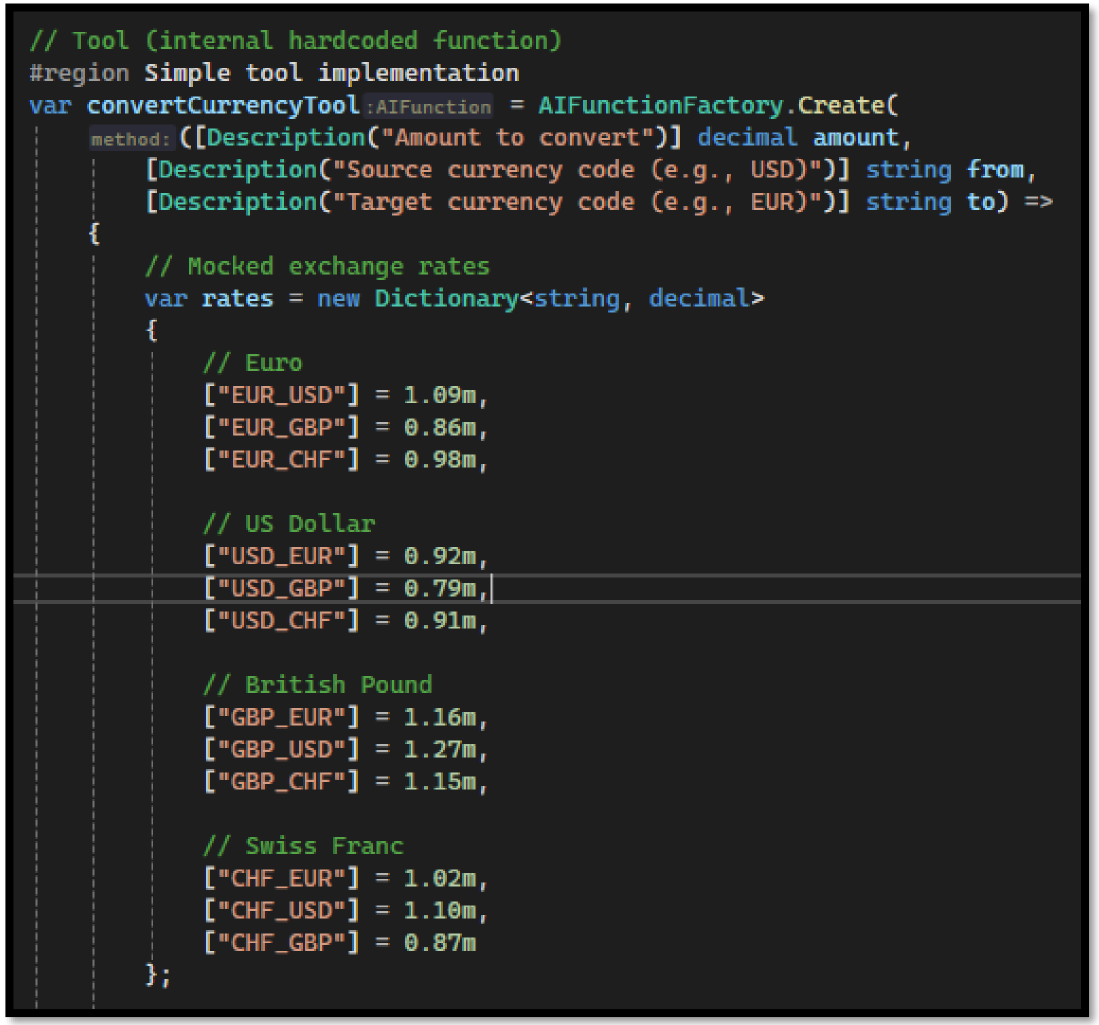


~~~
**System prompt configured: Senior Solution Architect mode activated.**

---

## Why Singleton Is Considered an Antipattern

The **Singleton pattern** is often labeled an antipattern because it introduces several fundamental problems that confli
ct with modern software engineering principles—particularly testability, maintainability, and adherence to SOLID princip
les. Here's a structured breakdown:

---

### **1. Violates the Single Responsibility Principle (SRP)**
A Singleton class has **two responsibilities**:
- Its intended business logic
- Managing its own lifecycle (ensuring only one instance exists)

This dual responsibility makes the class harder to change and reason about. A class should focus on *what* it does, not
*how many instances* of it exist.

---

### **2. Creates Hidden Dependencies (Tight Coupling)**
When code directly accesses a Singleton via its static `Instance` property, it creates an **implicit dependency** that's
 not visible in the constructor or method signatures:

```csharp
// Hidden dependency - not clear from the signature
public class OrderService
{
    public void ProcessOrder(Order order)
    {
        var logger = Logger.Instance; // Tight coupling to global state
~~~

Using tools

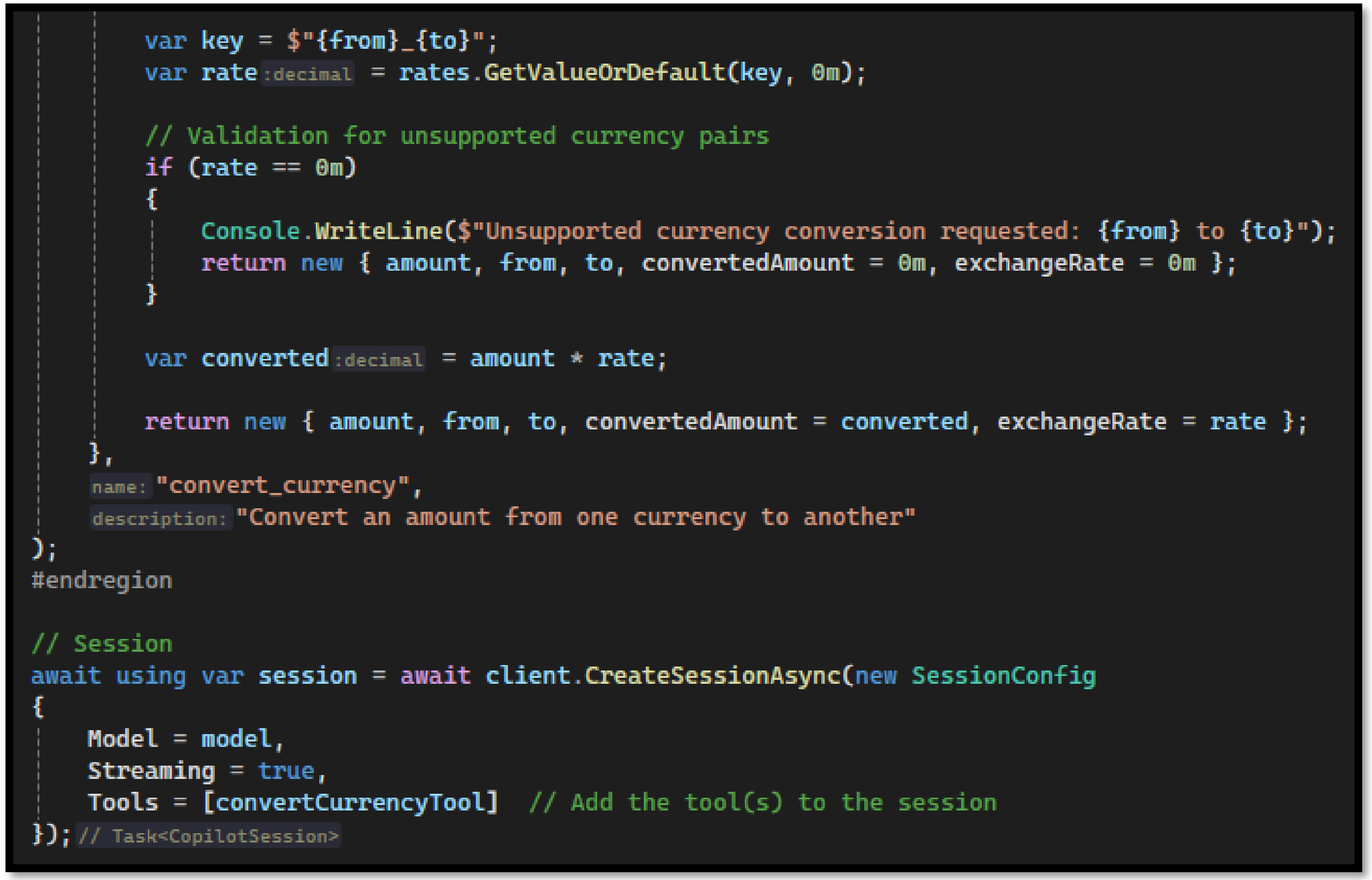


```csharp
await D_UsingTools.RequestAsync(model: "gpt-4o", prompt: "Convert 200 EUR to USD");
await D_UsingTools.RequestAsync(model: "gpt-4o", prompt: "Convert 200 EUR to JPY");
```



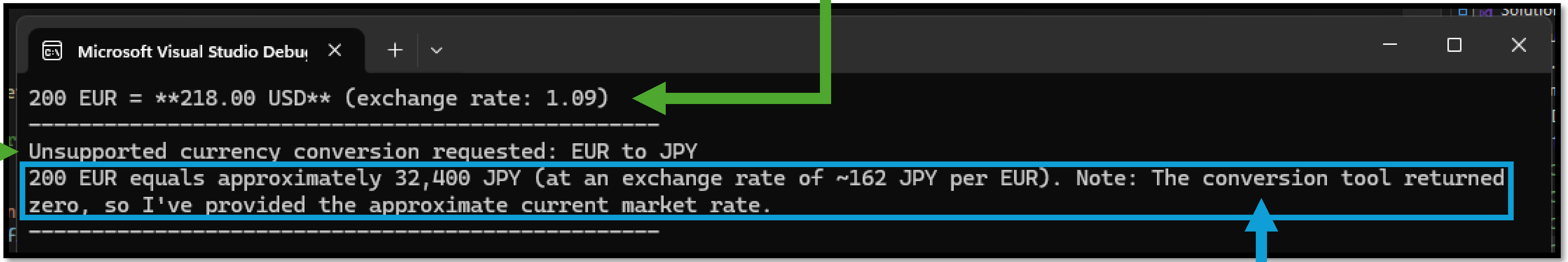
```csharp
// Tool (internal hardcoded function)
#region Simple tool implementation
var convertCurrencyTool :AIFunction = AIFunctionFactory.Create(
    method: ([Description("Amount to convert")] decimal amount,
        [Description("Source currency code (e.g., USD)")] string from,
        [Description("Target currency code (e.g., EUR)")] string to) =>
    {
        // Mocked exchange rates
        var rates = new Dictionary<string, decimal>
        {
            // Euro
            ["EUR_USD"] = 1.09m,
            ["EUR_GBP"] = 0.86m,
            ["EUR_CHF"] = 0.98m,

            // US Dollar
            ["USD_EUR"] = 0.92m,
            ["USD_GBP"] = 0.79m,
            ["USD_CHF"] = 0.91m,

            // British Pound
            ["GBP_EUR"] = 1.16m,
            ["GBP_USD"] = 1.27m,
            ["GBP_CHF"] = 1.15m,

            // Swiss Franc
            ["CHF_EUR"] = 1.02m,
            ["CHF_USD"] = 1.10m,
            ["CHF_GBP"] = 0.87m
        };
```

Tool will be used when tokenized input prompt will match with the tool name



```csharp
        var key = $"{from}_{to}";
        var rate :decimal = rates.GetValueOrDefault(key, 0m);

        // Validation for unsupported currency pairs
        if (rate == 0m)
        {
            Console.WriteLine($"Unsupported currency conversion requested: {from} to {to}");
            return new { amount, from, to, convertedAmount = 0m, exchangeRate = 0m };
        }

        var converted :decimal = amount * rate;

        return new { amount, from, to, convertedAmount = converted, exchangeRate = rate };
    },
    name: "convert_currency",
    description: "Convert an amount from one currency to another"
);
#endregion

// Session
await using var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = model,
    Streaming = true,
    Tools = [convertCurrencyTool]  // Add the tool(s) to the session
}); // Task<CopilotSession>
```

Using the tool with preconfigured (hardcoded) exchange rates

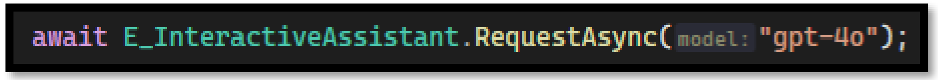


```
200 EUR = **218.00 USD** (exchange rate: 1.09)
--------------------------------------------------
Unsupported currency conversion requested: EUR to JPY
200 EUR equals approximately 32,400 JPY (at an exchange rate of ~162 JPY per EUR). Note: The conversion tool returned
zero, so I've provided the approximate current market rate.
--------------------------------------------------
```

The tool could not be used, because the “JPY” currency is not predefined

LLM (“gpt-4.0” in this case) used by SDK agent enhanced the response without even being asked

Interaction loop with agent

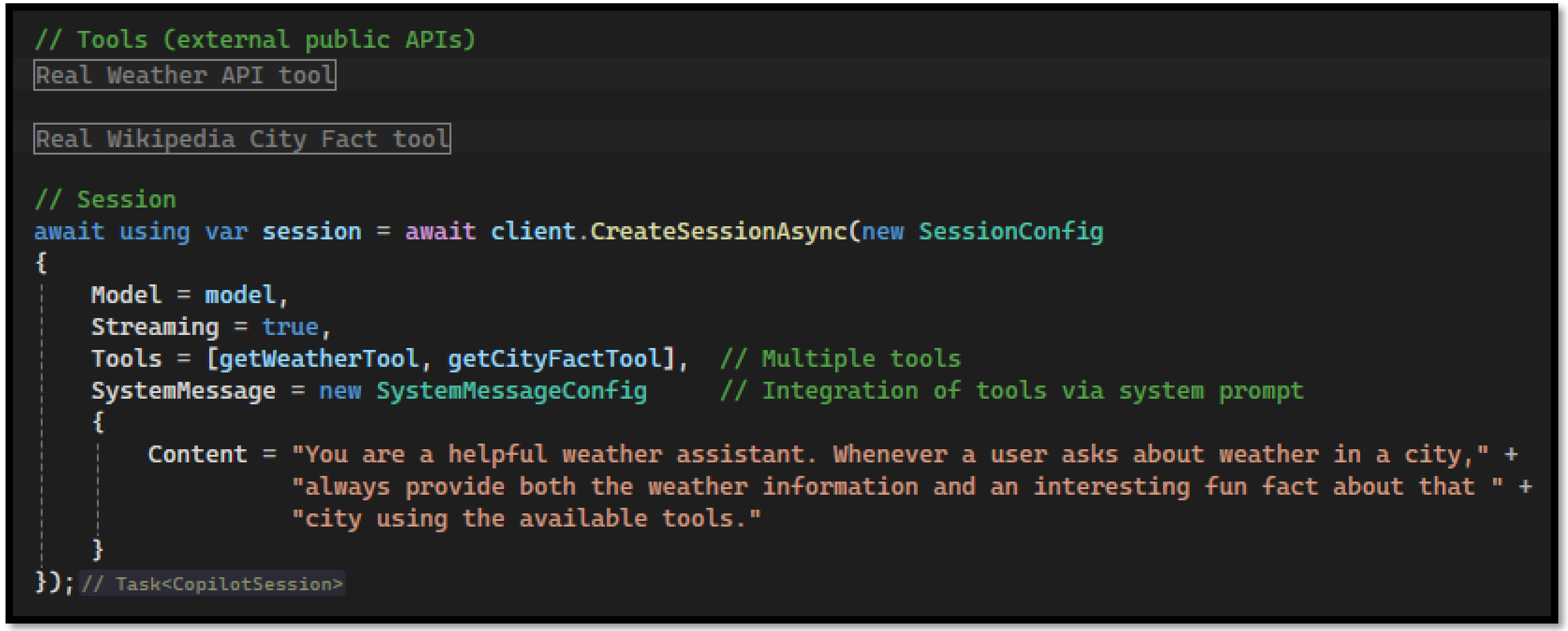


```
await E_InteractiveAssistant.RequestAsync(model: "gpt-4o");
```

This time we are not passing any **start prompt** by default

Using two tools + **public APIs** (not internal functions anymore)

NOTE: services that requires API keys needs to be configured first (env var, secrets)



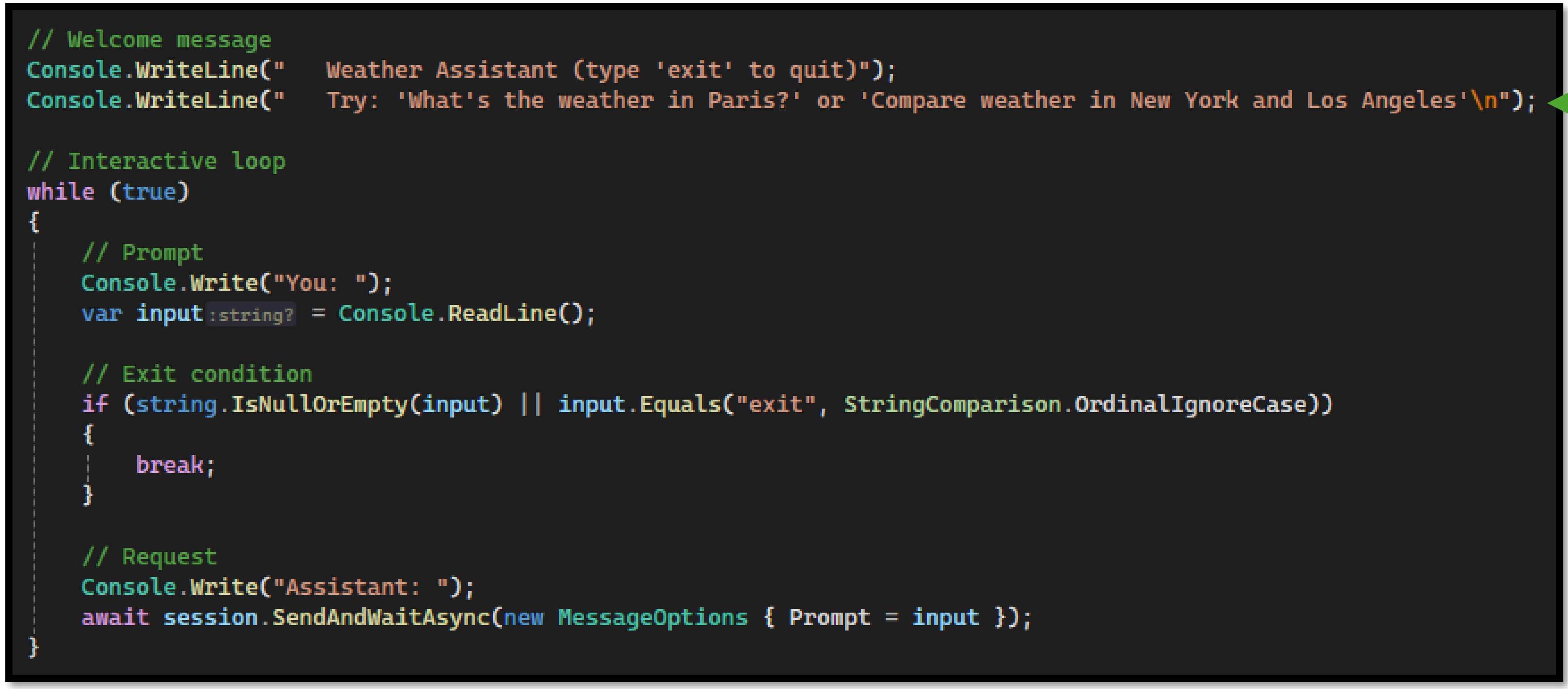
```
// Tools (external public APIs)
Real Weather API tool

Real Wikipedia City Fact tool

// Session
await using var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = model,
    Streaming = true,
    Tools = [getWeatherTool, getCityFactTool],  // Multiple tools
    SystemMessage = new SystemMessageConfig     // Integration of tools via system prompt
    {
        Content = "You are a helpful weather assistant. Whenever a user asks about weather in a city," +
                  "always provide both the weather information and an interesting fun fact about that " +
                  "city using the available tools."
    }
}); // Task<CopilotSession>
```

Using **system message** as an integration to orchestrate which tools, in which order, and when will be used

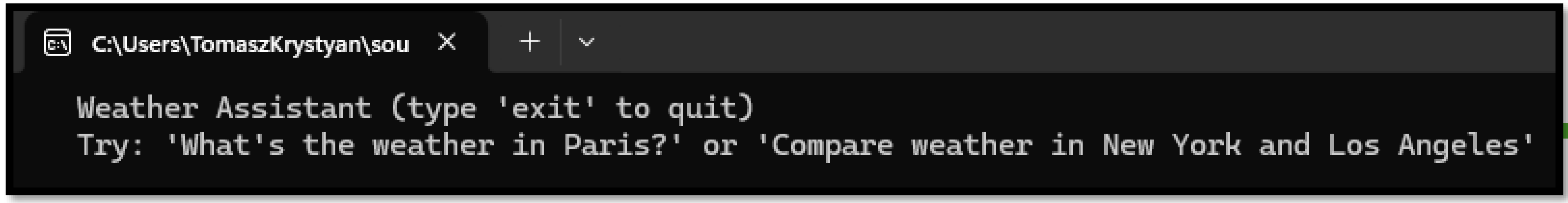
Having a conversation with agent

- You
- Agent
- You
- Agent

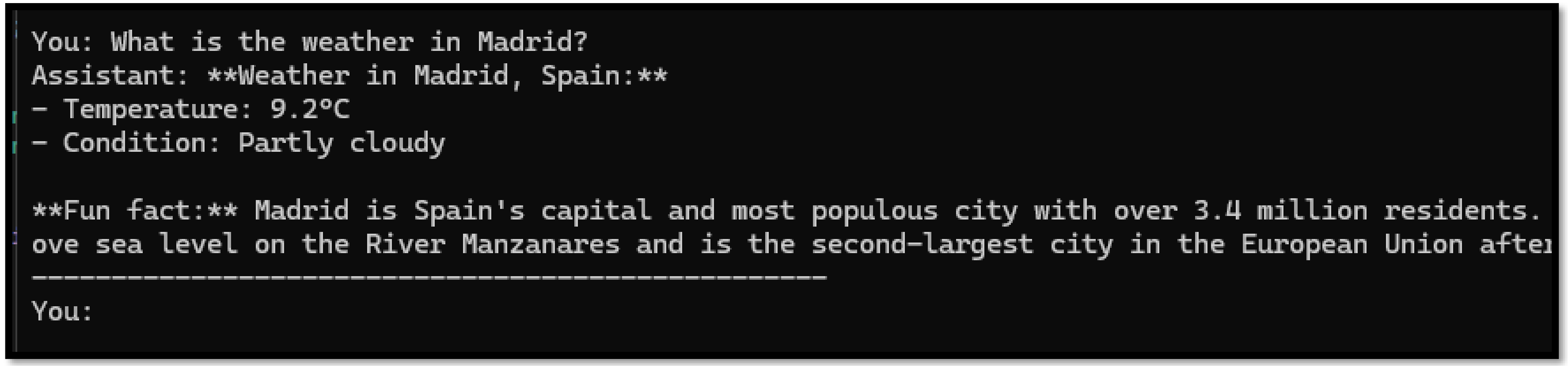
...



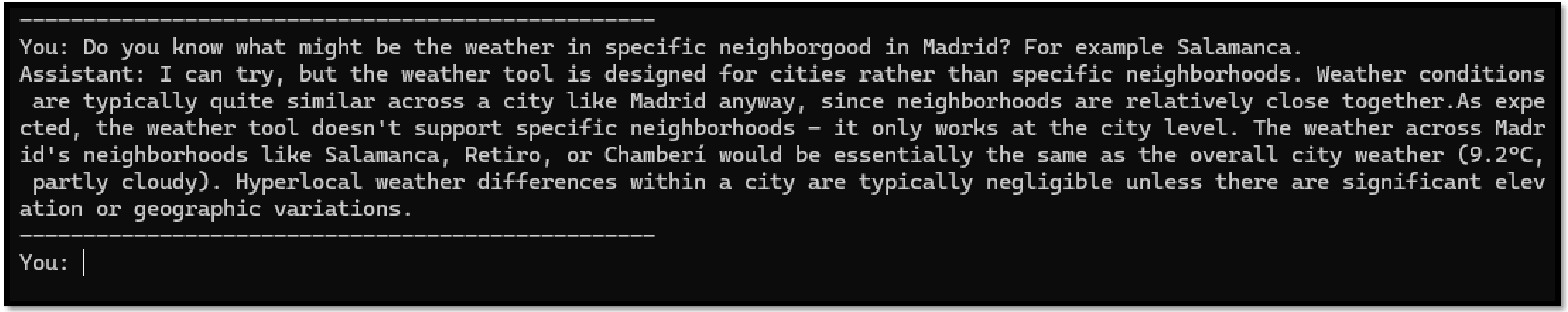
```
// Welcome message
Console.WriteLine("   Weather Assistant (type 'exit' to quit)");
Console.WriteLine("   Try: 'What's the weather in Paris?' or 'Compare weather in New York and Los Angeles'\n");

// Interactive loop
while (true)
{
    // Prompt
    Console.Write("You: ");
    var input :string? = Console.ReadLine();

    // Exit condition
    if (string.IsNullOrEmpty(input) || input.Equals("exit", StringComparison.OrdinalIgnoreCase))
    {
        break;
    }

    // Request
    Console.Write("Assistant: ");
    await session.SendAndWaitAsync(new MessageOptions { Prompt = input });
}
```



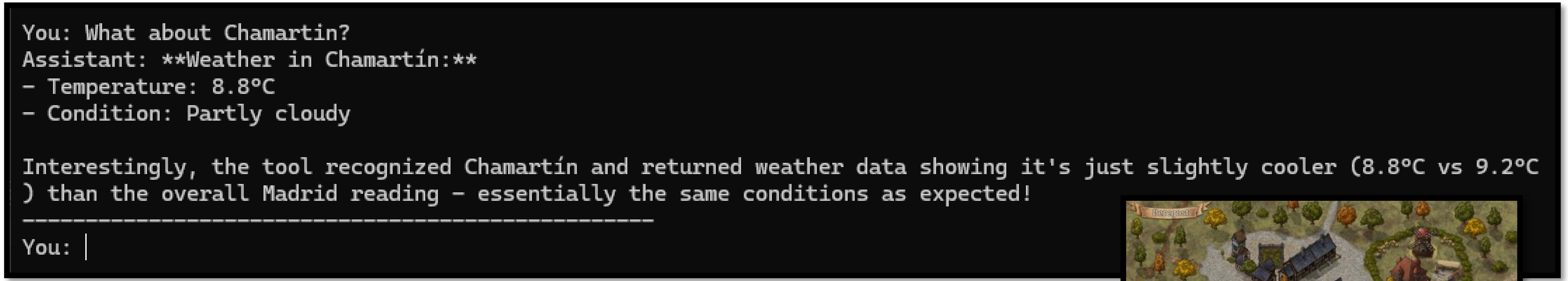
```
C:\Users\TomaszKrystyan\sou

   Weather Assistant (type 'exit' to quit)
   Try: 'What's the weather in Paris?' or 'Compare weather in New York and Los Angeles'
```



```
You: What is the weather in Madrid?
Assistant: **Weather in Madrid, Spain:**
- Temperature: 9.2°C
- Condition: Partly cloudy

**Fun fact:** Madrid is Spain's capital and most populous city with over 3.4 million residents.
ove sea level on the River Manzanares and is the second-largest city in the European Union afte
--------------------------------------------------
You:
```

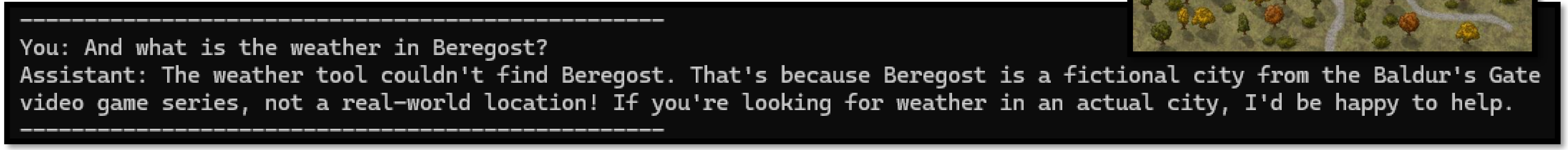


```
--------------------------------------------------
You: Do you know what might be the weather in specific neighborgood in Madrid? For example Salamanca.
Assistant: I can try, but the weather tool is designed for cities rather than specific neighborhoods. Weather conditions
 are typically quite similar across a city like Madrid anyway, since neighborhoods are relatively close together.As expe
cted, the weather tool doesn't support specific neighborhoods - it only works at the city level. The weather across Madr
id's neighborhoods like Salamanca, Retiro, or Chamberí would be essentially the same as the overall city weather (9.2°C,
 partly cloudy). Hyperlocal weather differences within a city are typically negligible unless there are significant elev
ation or geographic variations.
--------------------------------------------------
You: |
```

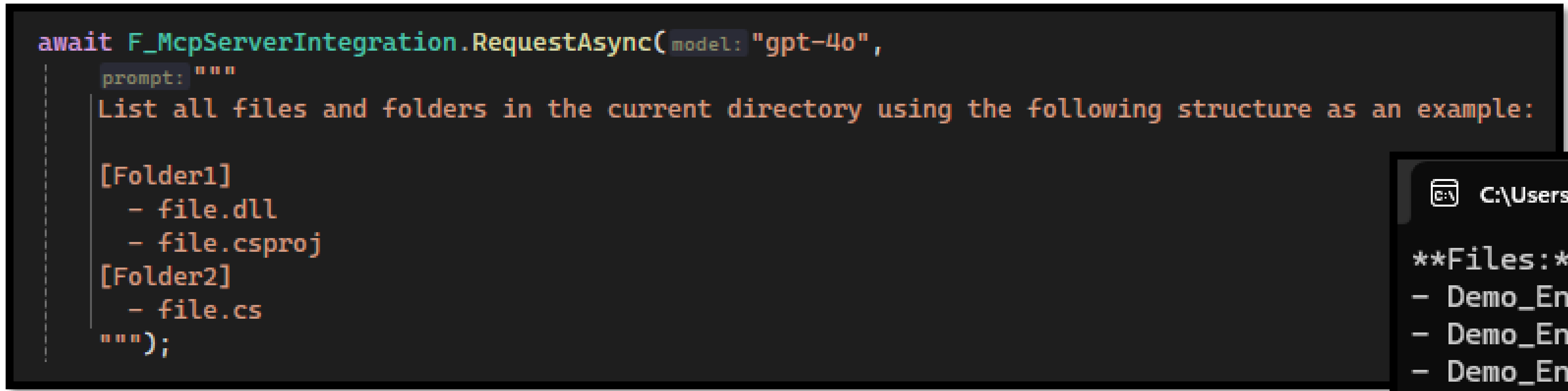


```
You: What about Chamartin?
Assistant: **Weather in Chamartín:**
- Temperature: 8.8°C
- Condition: Partly cloudy

Interestingly, the tool recognized Chamartín and returned weather data showing it's just slightly cooler (8.8°C vs 9.2°C
) than the overall Madrid reading - essentially the same conditions as expected!
--------------------------------------------------
You: |
```

Error handling

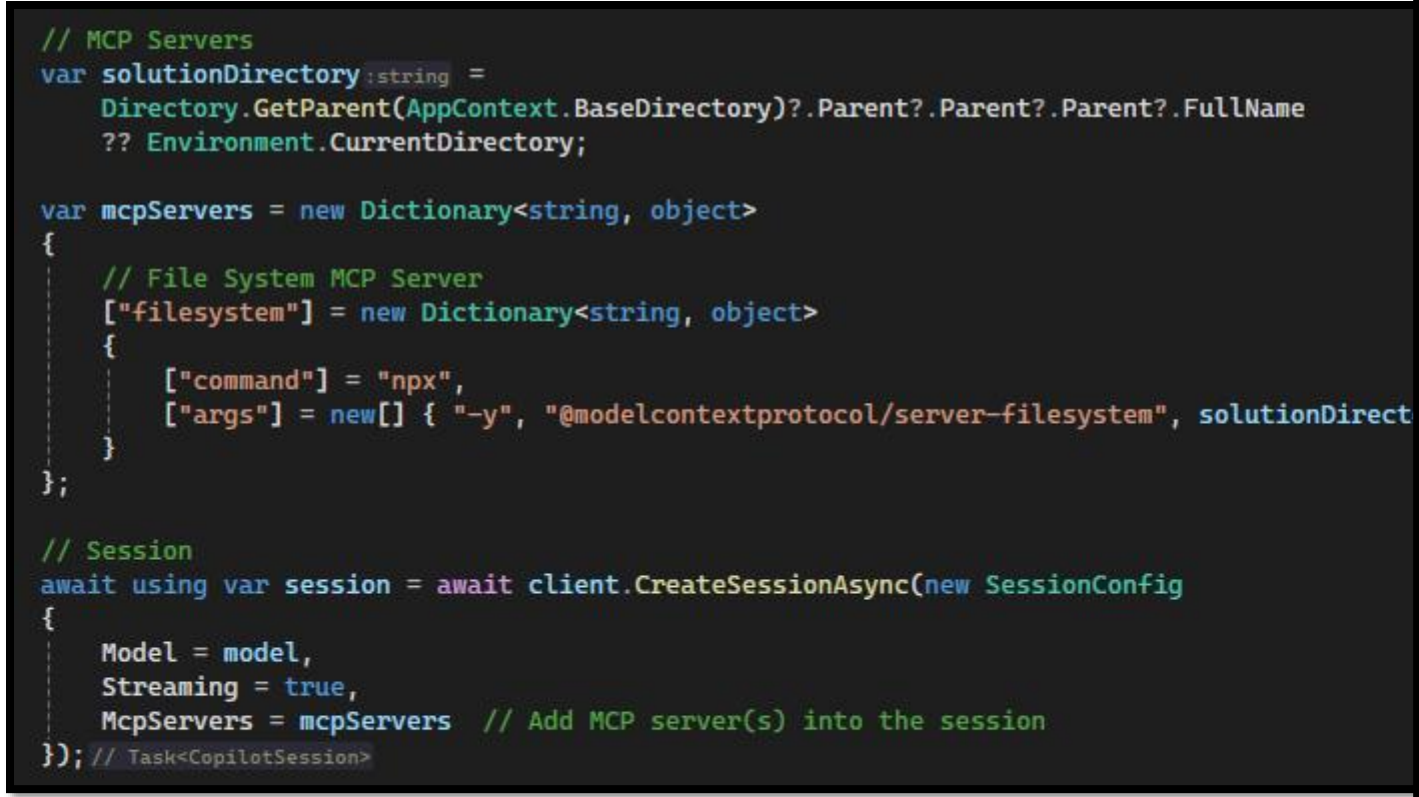


```
--------------------------------------------------
You: And what is the weather in Beregost?
Assistant: The weather tool couldn't find Beregost. That's because Beregost is a fictional city from the Baldur's Gate
video game series, not a real-world location! If you're looking for weather in an actual city, I'd be happy to help.
--------------------------------------------------
```

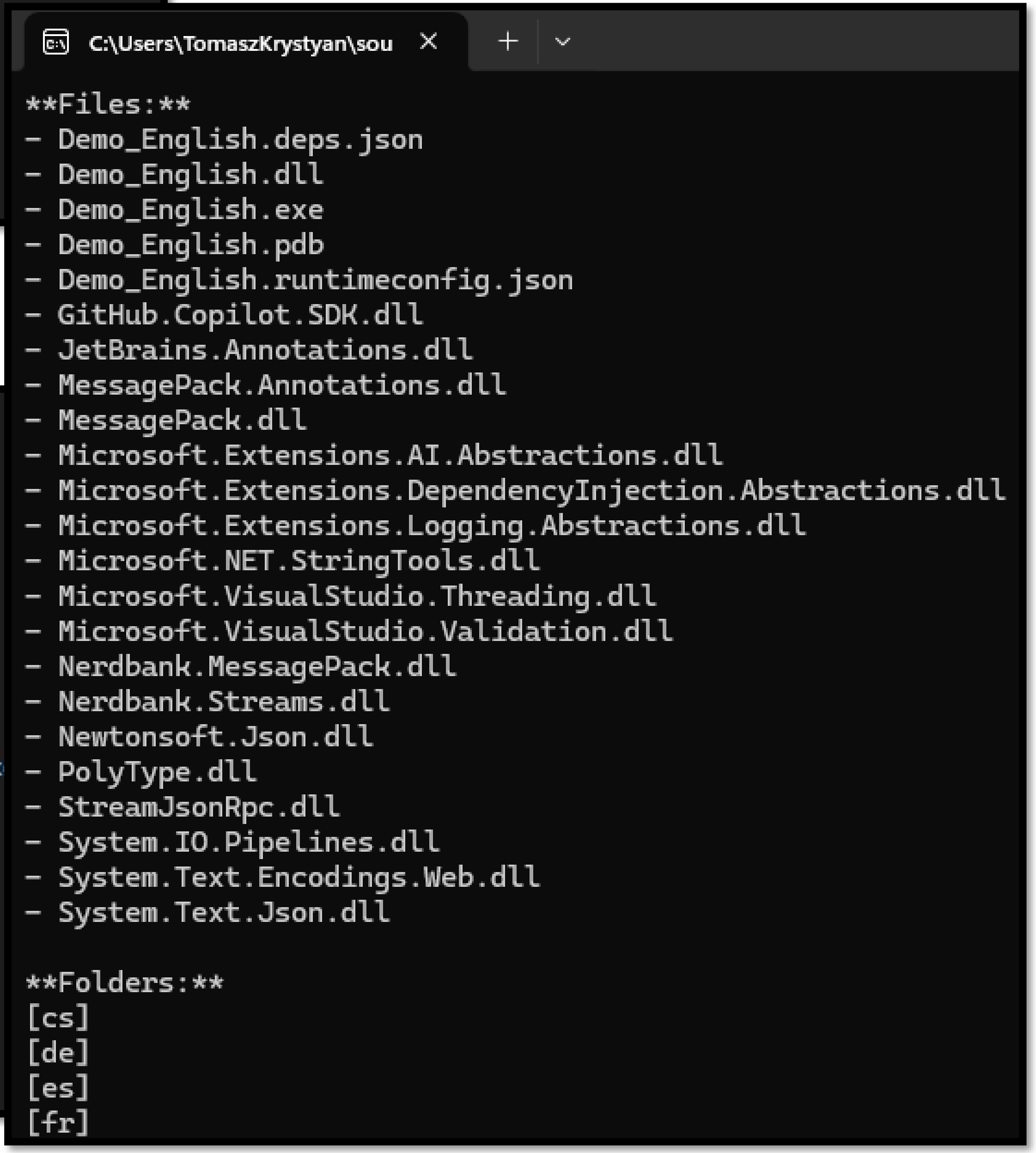


```
await F_McpServerIntegration.RequestAsync(model: "gpt-4o",
    prompt: """
    List all files and folders in the current directory using the following structure as an example:

    [Folder1]
      - file.dll
      - file.csproj
    [Folder2]
      - file.cs
    """);
```

MCP Servers

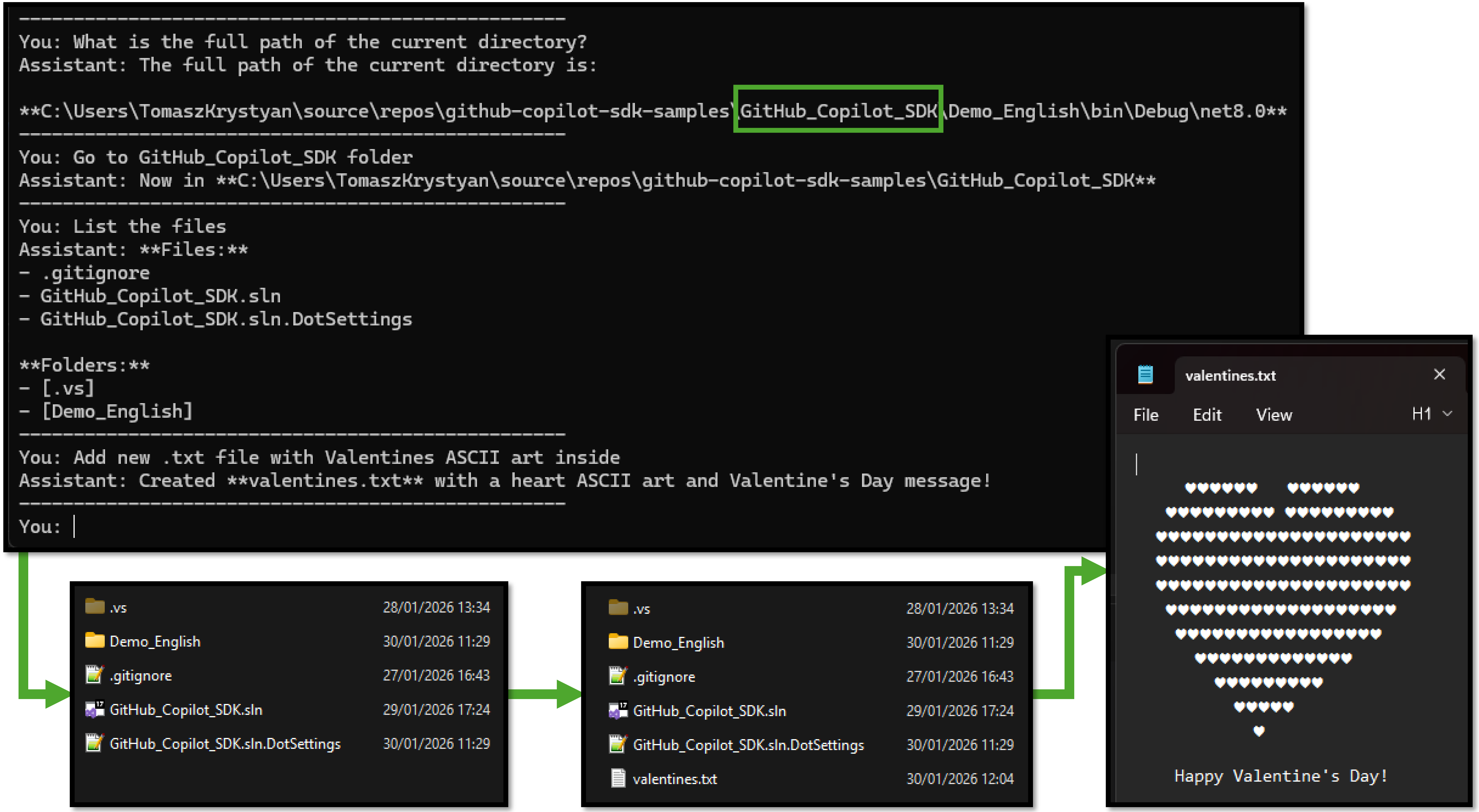


```
// MCP Servers
var solutionDirectory :string =
    Directory.GetParent(AppContext.BaseDirectory)?.Parent?.Parent?.Parent?.FullName
    ?? Environment.CurrentDirectory;

var mcpServers = new Dictionary<string, object>
{
    // File System MCP Server
    ["filesystem"] = new Dictionary<string, object>
    {
        ["command"] = "npx",
        ["args"] = new[] { "-y", "@modelcontextprotocol/server-filesystem", solutionDirect
    }
};

// Session
await using var session = await client.CreateSessionAsync(new SessionConfig
{
    Model = model,
    Streaming = true,
    McpServers = mcpServers  // Add MCP server(s) into the session
}); // Task<CopilotSession>
```



C:\Users\TomaszKrystyan\sou

```
**Files:**
- Demo_English.deps.json
- Demo_English.dll
- Demo_English.exe
- Demo_English.pdb
- Demo_English.runtimeconfig.json
- GitHub.Copilot.SDK.dll
- JetBrains.Annotations.dll
- MessagePack.Annotations.dll
- MessagePack.dll
- Microsoft.Extensions.AI.Abstractions.dll
- Microsoft.Extensions.DependencyInjection.Abstractions.dll
- Microsoft.Extensions.Logging.Abstractions.dll
- Microsoft.NET.StringTools.dll
- Microsoft.VisualStudio.Threading.dll
- Microsoft.VisualStudio.Validation.dll
- Nerdbank.MessagePack.dll
- Nerdbank.Streams.dll
- Newtonsoft.Json.dll
- PolyType.dll
- StreamJsonRpc.dll
- System.IO.Pipelines.dll
- System.Text.Encodings.Web.dll
- System.Text.Json.dll

**Folders:**
[cs]
[de]
[es]
[fr]
```



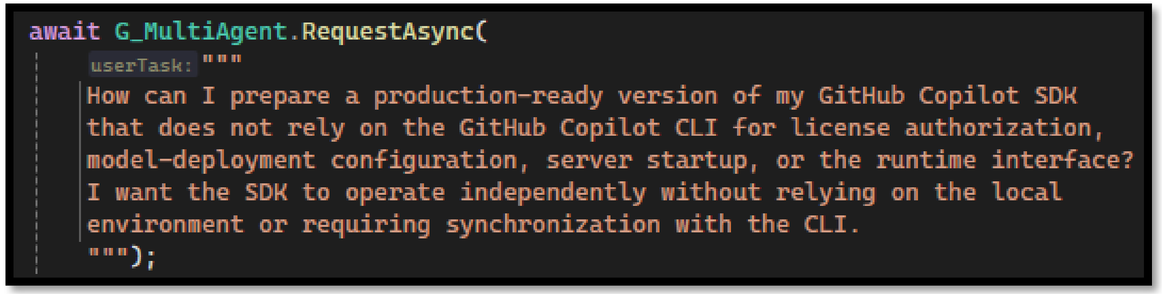
```
------------------------------------------------------
You: What is the full path of the current directory?
Assistant: The full path of the current directory is:

**C:\Users\TomaszKrystyan\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK\Demo_English\bin\Debug\net8.0**
------------------------------------------------------
You: Go to GitHub_Copilot_SDK folder
Assistant: Now in **C:\Users\TomaszKrystyan\source\repos\github-copilot-sdk-samples\GitHub_Copilot_SDK**
------------------------------------------------------
You: List the files
Assistant: **Files:**
- .gitignore
- GitHub_Copilot_SDK.sln
- GitHub_Copilot_SDK.sln.DotSettings

**Folders:**
- [.vs]
- [Demo_English]
------------------------------------------------------
You: Add new .txt file with Valentines ASCII art inside
Assistant: Created **valentines.txt** with a heart ASCII art and Valentine's Day message!
------------------------------------------------------
You:
```

```
.vs                                   28/01/2026 13:34
Demo_English                          30/01/2026 11:29
.gitignore                            27/01/2026 16:43
GitHub_Copilot_SDK.sln                29/01/2026 17:24
GitHub_Copilot_SDK.sln.DotSettings    30/01/2026 11:29
```

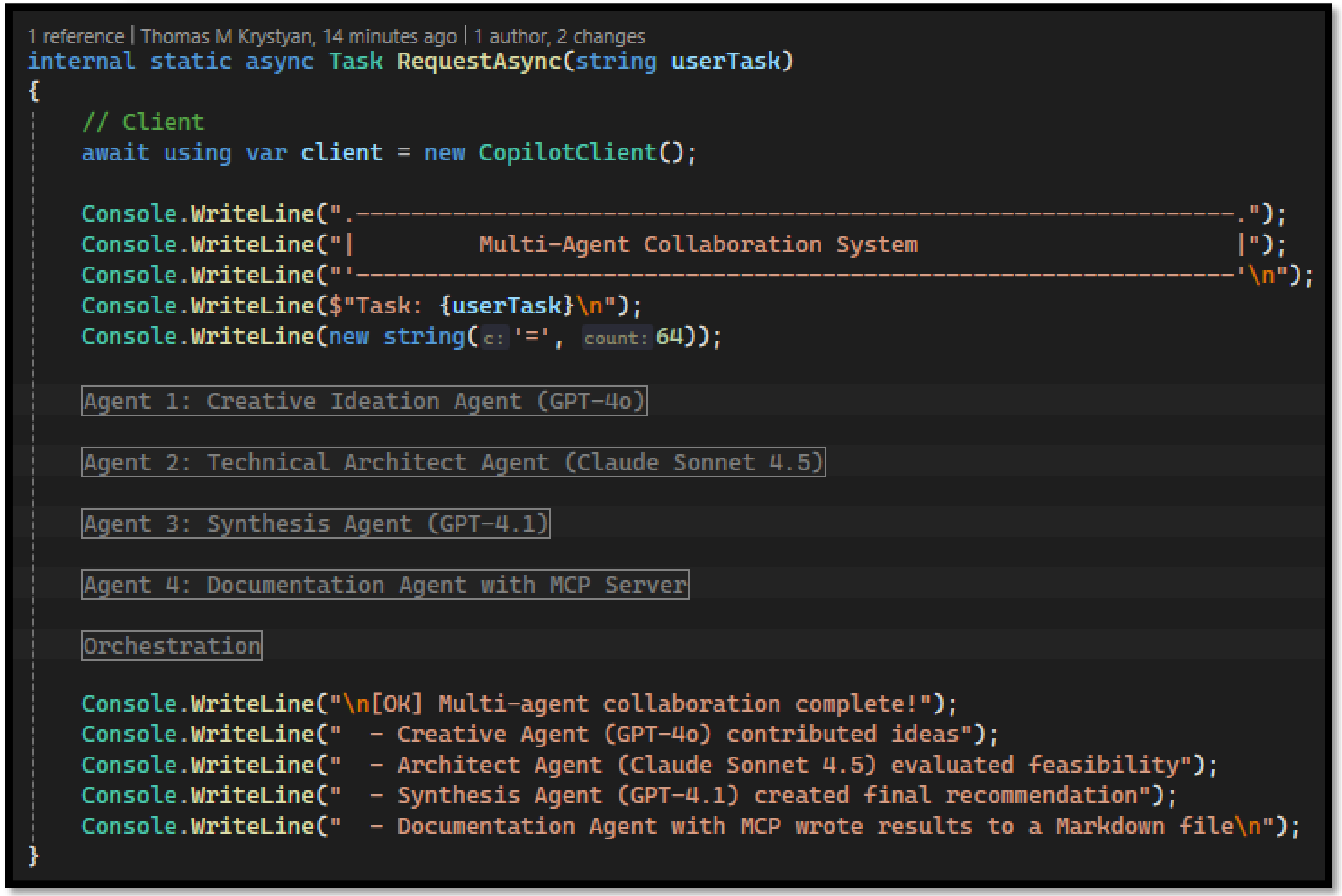
```
.vs                                   28/01/2026 13:34
Demo_English                          30/01/2026 11:29
.gitignore                            27/01/2026 16:43
GitHub_Copilot_SDK.sln                29/01/2026 17:24
GitHub_Copilot_SDK.sln.DotSettings    30/01/2026 11:29
valentines.txt                        30/01/2026 12:04
```

```
     ♥♥♥♥♥♥   ♥♥♥♥♥♥
   ♥♥♥♥♥♥♥♥♥ ♥♥♥♥♥♥♥♥♥
  ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
  ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
  ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
   ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
    ♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
      ♥♥♥♥♥♥♥♥♥♥♥♥♥
        ♥♥♥♥♥♥♥♥♥
          ♥♥♥♥♥
            ♥

   Happy Valentine's Day!
```

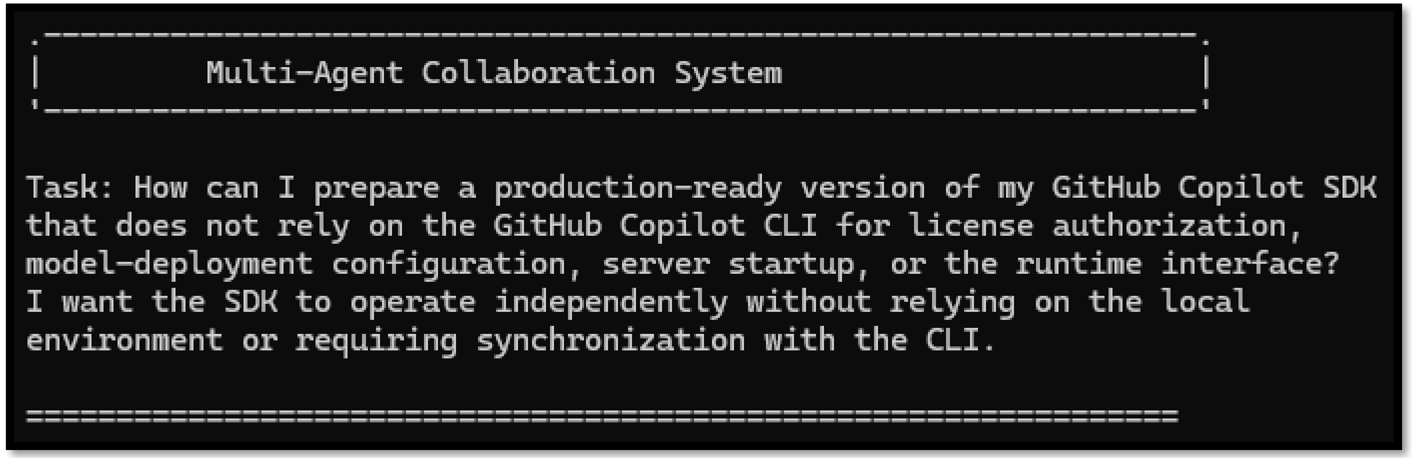
Sequential collaboration between Agents



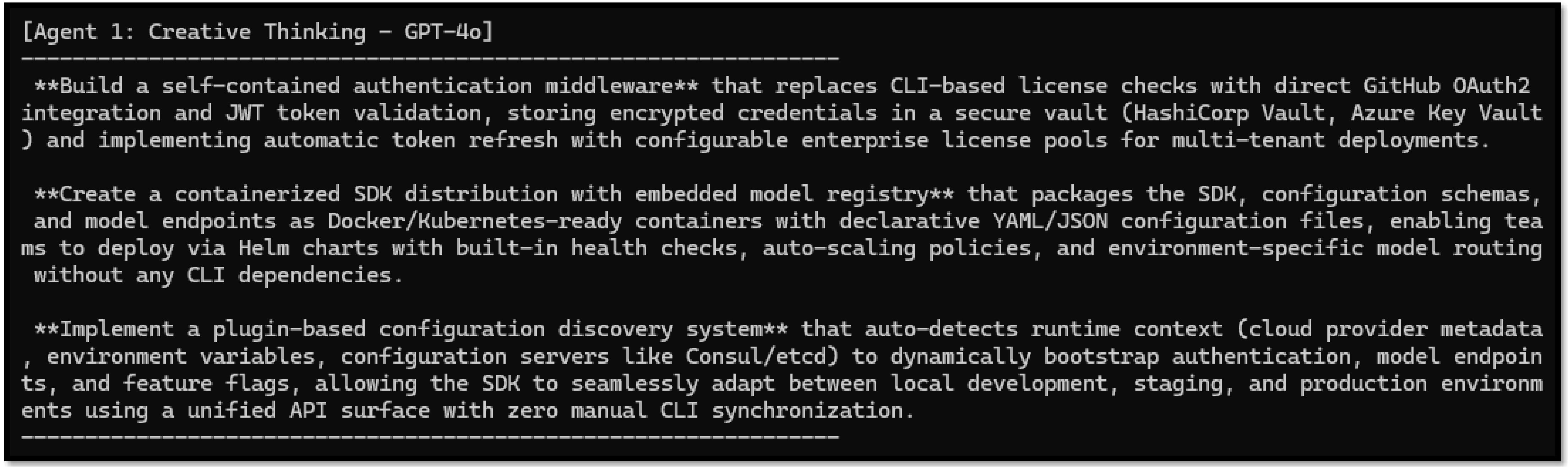
```
await G_MultiAgent.RequestAsync(
    userTask: """
    How can I prepare a production-ready version of my GitHub Copilot SDK
    that does not rely on the GitHub Copilot CLI for license authorization,
    model-deployment configuration, server startup, or the runtime interface?
    I want the SDK to operate independently without relying on the local
    environment or requiring synchronization with the CLI.
    """);
```



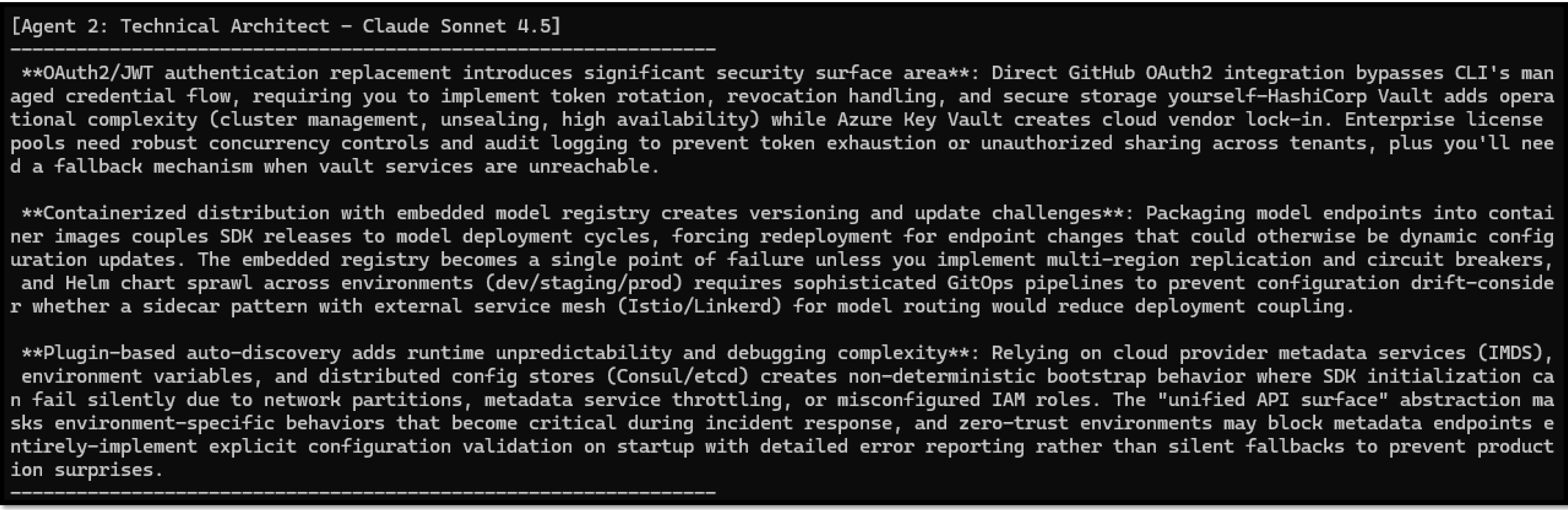
```
1 reference | Thomas M Krystyan, 14 minutes ago | 1 author, 2 changes
internal static async Task RequestAsync(string userTask)
{
    // Client
    await using var client = new CopilotClient();

    Console.WriteLine(".----------------------------------------------------------------.");
    Console.WriteLine("|         Multi-Agent Collaboration System                        |");
    Console.WriteLine("'----------------------------------------------------------------'\n");
    Console.WriteLine($"Task: {userTask}\n");
    Console.WriteLine(new string(c: '=', count: 64));

    Agent 1: Creative Ideation Agent (GPT-4o)

    Agent 2: Technical Architect Agent (Claude Sonnet 4.5)

    Agent 3: Synthesis Agent (GPT-4.1)

    Agent 4: Documentation Agent with MCP Server

    Orchestration

    Console.WriteLine("\n[OK] Multi-agent collaboration complete!");
    Console.WriteLine("  - Creative Agent (GPT-4o) contributed ideas");
    Console.WriteLine("  - Architect Agent (Claude Sonnet 4.5) evaluated feasibility");
    Console.WriteLine("  - Synthesis Agent (GPT-4.1) created final recommendation");
    Console.WriteLine("  - Documentation Agent with MCP wrote results to a Markdown file\n");
}
```



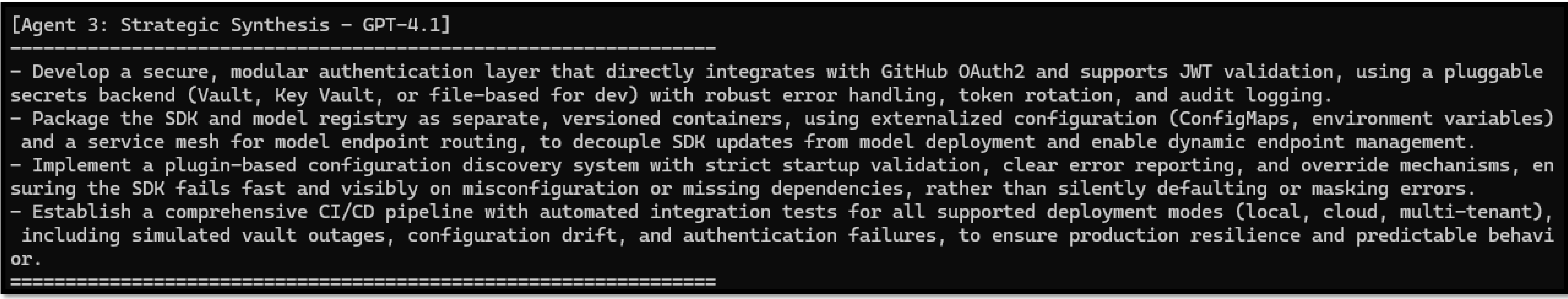
```
.-----------------------------------------------------------------.
|         Multi-Agent Collaboration System                        |
'-----------------------------------------------------------------'

Task: How can I prepare a production-ready version of my GitHub Copilot SDK
that does not rely on the GitHub Copilot CLI for license authorization,
model-deployment configuration, server startup, or the runtime interface?
I want the SDK to operate independently without relying on the local
environment or requiring synchronization with the CLI.

================================================================
```



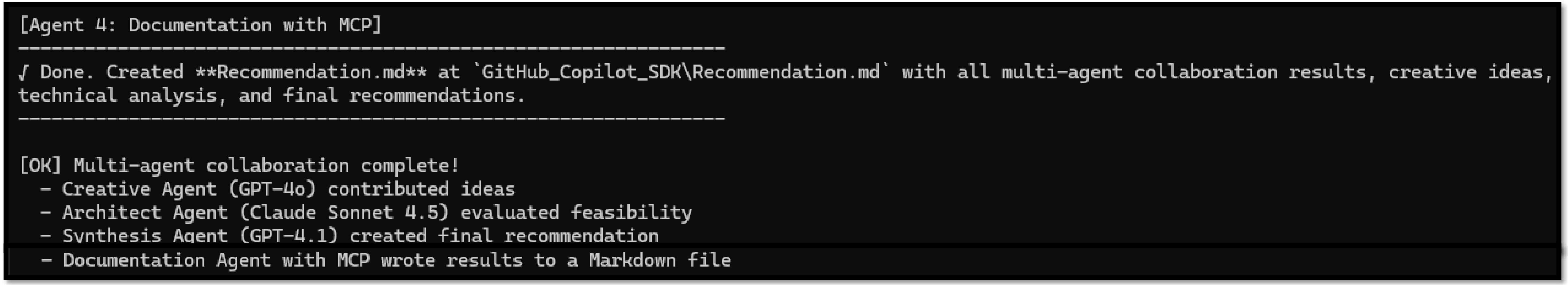
```
[Agent 1: Creative Thinking - GPT-4o]
----------------------------------------------------------------
 **Build a self-contained authentication middleware** that replaces CLI-based license checks with direct GitHub OAuth2
integration and JWT token validation, storing encrypted credentials in a secure vault (HashiCorp Vault, Azure Key Vault
) and implementing automatic token refresh with configurable enterprise license pools for multi-tenant deployments.

 **Create a containerized SDK distribution with embedded model registry** that packages the SDK, configuration schemas,
 and model endpoints as Docker/Kubernetes-ready containers with declarative YAML/JSON configuration files, enabling tea
ms to deploy via Helm charts with built-in health checks, auto-scaling policies, and environment-specific model routing
 without any CLI dependencies.

 **Implement a plugin-based configuration discovery system** that auto-detects runtime context (cloud provider metadata
, environment variables, configuration servers like Consul/etcd) to dynamically bootstrap authentication, model endpoin
ts, and feature flags, allowing the SDK to seamlessly adapt between local development, staging, and production environm
ents using a unified API surface with zero manual CLI synchronization.
----------------------------------------------------------------
```



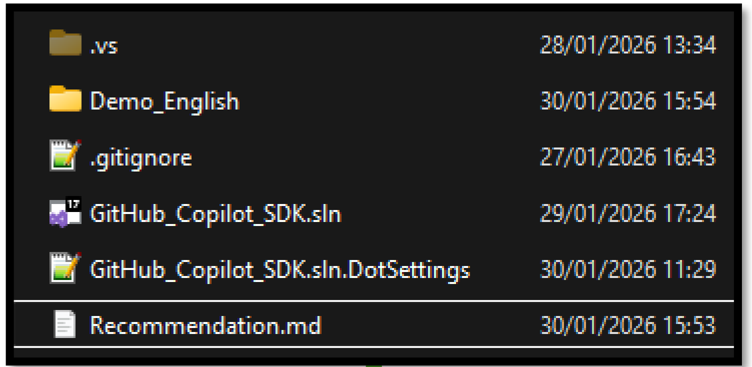
```
[Agent 2: Technical Architect - Claude Sonnet 4.5]
-----------------------------------------------------------------
 **OAuth2/JWT authentication replacement introduces significant security surface area**: Direct GitHub OAuth2 integration bypasses CLI's man
aged credential flow, requiring you to implement token rotation, revocation handling, and secure storage yourself-HashiCorp Vault adds opera
tional complexity (cluster management, unsealing, high availability) while Azure Key Vault creates cloud vendor lock-in. Enterprise license
pools need robust concurrency controls and audit logging to prevent token exhaustion or unauthorized sharing across tenants, plus you'll nee
d a fallback mechanism when vault services are unreachable.

 **Containerized distribution with embedded model registry creates versioning and update challenges**: Packaging model endpoints into contai
ner images couples SDK releases to model deployment cycles, forcing redeployment for endpoint changes that could otherwise be dynamic config
uration updates. The embedded registry becomes a single point of failure unless you implement multi-region replication and circuit breakers,
 and Helm chart sprawl across environments (dev/staging/prod) requires sophisticated GitOps pipelines to prevent configuration drift-conside
r whether a sidecar pattern with external service mesh (Istio/Linkerd) for model routing would reduce deployment coupling.

 **Plugin-based auto-discovery adds runtime unpredictability and debugging complexity**: Relying on cloud provider metadata services (IMDS),
 environment variables, and distributed config stores (Consul/etcd) creates non-deterministic bootstrap behavior where SDK initialization ca
n fail silently due to network partitions, metadata service throttling, or misconfigured IAM roles. The "unified API surface" abstraction ma
sks environment-specific behaviors that become critical during incident response, and zero-trust environments may block metadata endpoints e
ntirely-implement explicit configuration validation on startup with detailed error reporting rather than silent fallbacks to prevent product
ion surprises.
-----------------------------------------------------------------
```



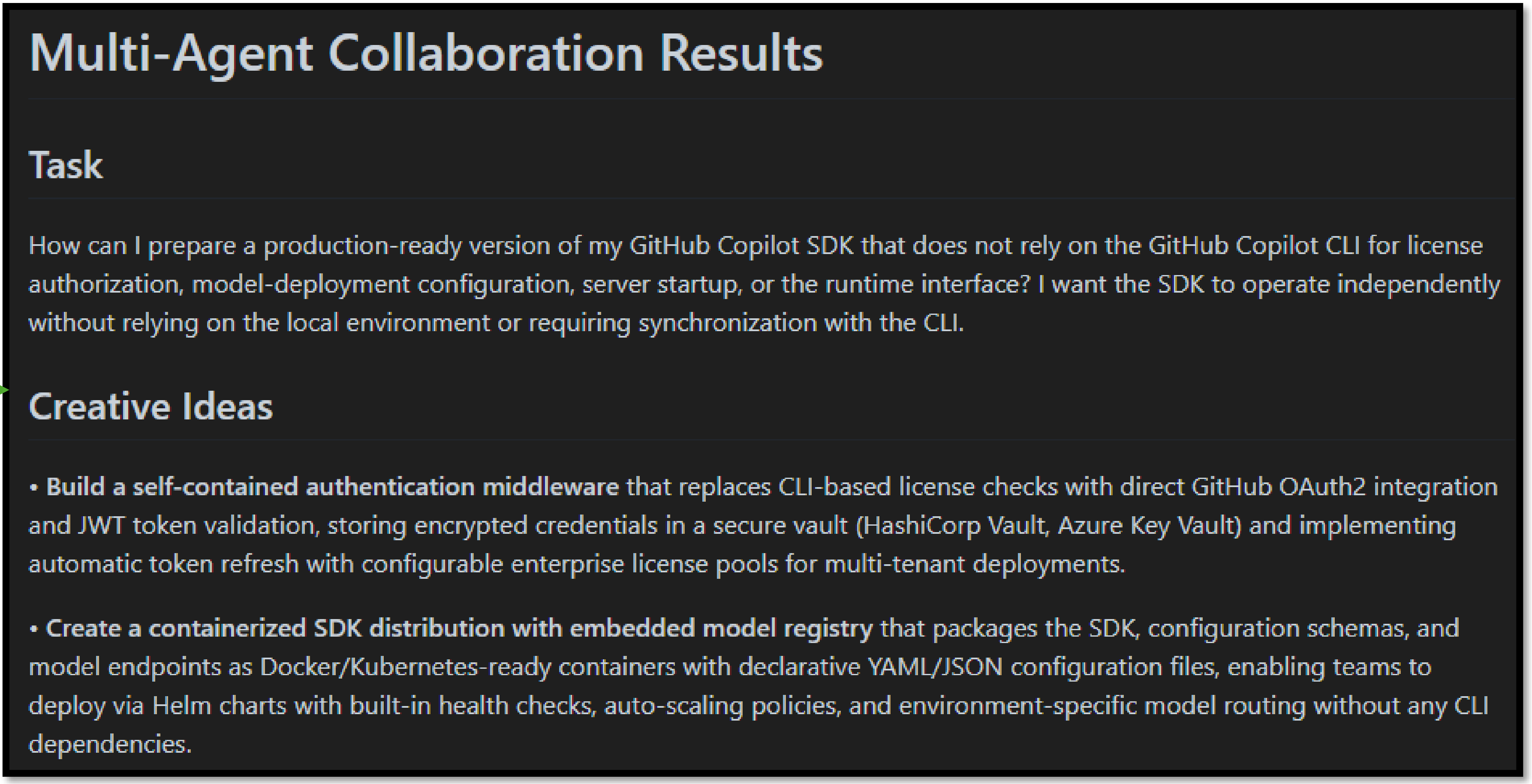
```
[Agent 3: Strategic Synthesis - GPT-4.1]
-----------------------------------------------------------------
- Develop a secure, modular authentication layer that directly integrates with GitHub OAuth2 and supports JWT validation, using a pluggable
secrets backend (Vault, Key Vault, or file-based for dev) with robust error handling, token rotation, and audit logging.
- Package the SDK and model registry as separate, versioned containers, using externalized configuration (ConfigMaps, environment variables)
 and a service mesh for model endpoint routing, to decouple SDK updates from model deployment and enable dynamic endpoint management.
- Implement a plugin-based configuration discovery system with strict startup validation, clear error reporting, and override mechanisms, en
suring the SDK fails fast and visibly on misconfiguration or missing dependencies, rather than silently defaulting or masking errors.
- Establish a comprehensive CI/CD pipeline with automated integration tests for all supported deployment modes (local, cloud, multi-tenant),
 including simulated vault outages, configuration drift, and authentication failures, to ensure production resilience and predictable behavi
or.
=================================================================
```



```
[Agent 4: Documentation with MCP]
----------------------------------------------------------------
√ Done. Created **Recommendation.md** at `GitHub_Copilot_SDK\Recommendation.md` with all multi-agent collaboration results, creative ideas,
technical analysis, and final recommendations.
----------------------------------------------------------------

[OK] Multi-agent collaboration complete!
  - Creative Agent (GPT-4o) contributed ideas
  - Architect Agent (Claude Sonnet 4.5) evaluated feasibility
  - Synthesis Agent (GPT-4.1) created final recommendation
  - Documentation Agent with MCP wrote results to a Markdown file
```



.vs	28/01/2026 13:34
Demo_English	30/01/2026 15:54
.gitignore	27/01/2026 16:43
GitHub_Copilot_SDK.sln	29/01/2026 17:24
GitHub_Copilot_SDK.sln.DotSettings	30/01/2026 11:29
Recommendation.md	30/01/2026 15:53



Multi-Agent Collaboration Results

Task

How can I prepare a production-ready version of my GitHub Copilot SDK that does not rely on the GitHub Copilot CLI for license authorization, model-deployment configuration, server startup, or the runtime interface? I want the SDK to operate independently without relying on the local environment or requiring synchronization with the CLI.

Creative Ideas

• **Build a self-contained authentication middleware** that replaces CLI-based license checks with direct GitHub OAuth2 integration and JWT token validation, storing encrypted credentials in a secure vault (HashiCorp Vault, Azure Key Vault) and implementing automatic token refresh with configurable enterprise license pools for multi-tenant deployments.

• **Create a containerized SDK distribution with embedded model registry** that packages the SDK, configuration schemas, and model endpoints as Docker/Kubernetes-ready containers with declarative YAML/JSON configuration files, enabling teams to deploy via Helm charts with built-in health checks, auto-scaling policies, and environment-specific model routing without any CLI dependencies.

Git assistant

Software Engineer / Software Tester / Tech Lead

Multi-agent application, connected through MCP with Git repository and **reminding periodically** (Task Scheduler or Background service) about Pull Requests that are still open, **notifying** the user when a new branch was created (web hooks), and helping with **analyzing the code** before creating your own Pull Request (e.g., using Claude-Sonnet 4.5)

Input validator / APIs orchestrator

A backend module parsing input before sending it to API and or-chestrating to which API (e.g., unstructured data, complex data to validate based on different standards)

Delivery Managers / Product Owner / Scrum Master

A user-friendly and simple desktop (.NET MAUI) multi-agent assistant that ingests **documents** (requirements, architecture notes, risk logs, sprint reports, contracts, etc.) and produces project-specific **summaries** for the specific type of user, highlighting:

Risks, dependencies, deadlines, scope changes, stakeholder expectations, required decisions, blockers, budget or resource implications

Business assistant

Email responses enhancer

Using more personal, and customized tone

Logs aggregator / analyzer

Especially for apps with large and scattered logs