**研究报告**

**课题名称：利用大模型和MCP打造智能助手**

**摘要：**在目前消费级的PC上，虽然MS的Windows占有大量的市场，但是缺少像Siri类似的底层桌面端AI Agent。该课题主要研究了把AI大模型作为处理和理解用户请求的工具，并将大模型做出的回答返回给用户的客户端。该课题也探究了如何将AI大模型实时编写的python代码直接运行在用户的电脑上。从而帮助用户完成操作或任务。

**关键词：人工智能；AI Agent；Python；大模型**

# 引言

**1.1课题的来源与背景**

我常常希望在使用桌面端PC时，可以像使用手机/平板等移动电子设备时，让设备代替我们部分手动的操控流程，来节省力气。就不如我需要打开某个游戏时可能会在文件紊乱的桌面上找不到它的快捷方式，也不想再次打开可能已经被我关闭的页面来寻找

MS的Windows Copilot支持如Windows Recall、Windows Studio Effects、实时字幕翻译、语音清晰度、自动超级分辨率、「画图」中的图像创建器、「照片」中的 Restyle 图像功能、Copilot 应用等等，但是这些都过度集中于部分软件应用程序，却没考虑到执行底层任务

**1.2课题研究的目的与意义**

该智能助手运用了MCP工具、大模型、API调用技术，实现主要功能是向大模型发送用户请求，大模型间接帮助用户完成其发布的指令，比如删除文件或者是移动文件到回收站，亦或者是进行搜索任务。减轻了如Siri这种本地模型在进行问答时对本地端设备性能造成压力的情况

**1.3查新情况**

通过在知网中下载并阅读文献，笔者将其与本课题进行了分析、对比，发现本课题的优在于：

1）便于在本地端快速部署，省去了额外的安装环节

2）使用API接入大模型的方式，减轻了本地模型在电脑上的运行压力

3）支持底层操作，如打开应用或者是打开指定的文件（非桌面文件需要路径）

# 并且在检索中未见与本课题相同的报道，表明目前还没有相似的课题，因而本课题的研究具有一定的价值和意义。

# 2、技术原理和总体思路

课题的主要功能包括：1.向大模型发送用户请求 2.让PC执行用户的要求或指令

课题采用API调用技术，实现向大模型发送用户请求功能，用MCP工具技术，实现让PC执行用户的要求或指令功能。。。

**2.1 API调用**

智能助手要实现与人类对话的效果，鉴于开发和维护专有 LLM 的成本很高，这些 AI 代理建立在 LLM 模型（可以是专有的）或 ChatGPT 等外部模型之上。可以使用Deep Seek或者是ChatGPT的API接口，虽然像 ChatGPT 这样的通用 LLM 模型由于拥有广泛的训练数据，在开放领域问答中表现良好，但与各种外部工具的集成对于涉及特定领域知识和任务执行的实际应用程序至关重要，这称为工具增强 LLM。例如，AI 代理必须与天气信息 API 集成以提供当地天气预报，或与预订 API 集成以处理住宿预订。

**2.2 MCP工具**

要让电脑本地的python运行由大模型生成的根据用户要求编写的代码，我们需要建立一个MCP工具来执行，把本地工具（如爬虫、Jupyter、API 接口、GitHub Repo）包装成“AI 能识别和调用”的服务。本身语言模型不具备实际能力去操作网页、读代码、调接口。 MCP 就像给它配了“远程遥控器”

**2.3 研究过程**

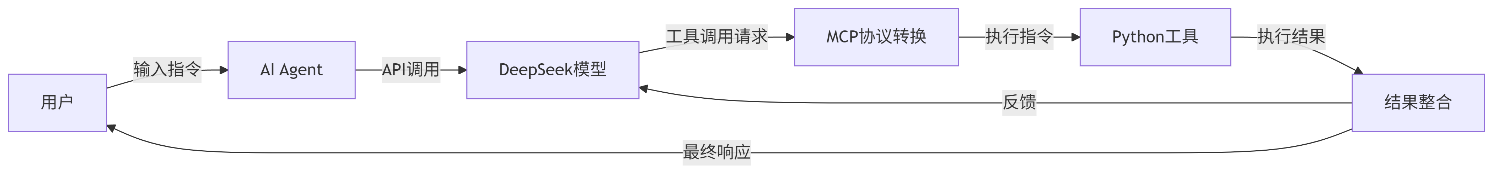
先用python进行了MCP工具的技术验证，实现了使用MCP工具运行代码的功能。

为了实用化部署和方便使用，用python完成了Deep Seek API的引入，实验表明用改变系统提示词的方法可以完成针对本地更加符合要求的命令

通过一系列的实验，对操作文件和查找内容以及打开应用功能进行了实验，得出了实验结论。

**2.4 总体技术思路**

让Deep Seek大模型作为用户请求的接收人，让大模型处理用户的自然语言并且分析需求，根据需求生成代码，或者是根据用户的关键词在网上搜索、寻找信息



图一：系统技术路线图

# 3、系统实现

项目用Deep Seek API实现对于用户请求内容的处理和分析，使用MCP协议，MCP的核心是**模型上下文**，即LLM在运行过程中所需的所有外部信息和工具。通过定义标准化的接口和协议，MCP使LLM能够动态访问和集成外部数据源、工具和服务。例如，LLM可以通过MCP访问数据库、API、文档库等外部数据源，或使用计算工具、搜索引擎、第三方服务等扩展其功能。

**3.1 系统代码实现**

* Python的实现

项目采用python进行开发，使用了python的OpenAI和MCP的 SDK，完成了基础人机交互功能，

系统代码分为4个部分，第一部分是环境配置模块，第二部分是系统提示模块，第三部分是工具转换模块，第四部分是核心运行模块

1. 环境配置代码：导入所有必要的库，设置 OpenAI API 密钥和基础 URL，配置 DeepSeek 模型参数，定义 MCP 服务器连接参数

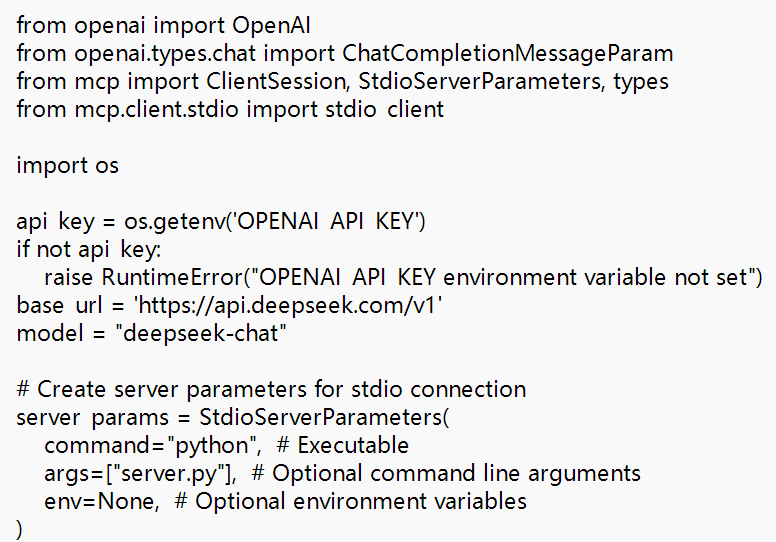


图1：系统初始化代码

1. 系统提示模块：该模块是系统提示词，定义 AI Agent 的行为准则（26 条注意事项），指定工具使用规范（Python/网络搜索），设置环境要求和预装库清单，提供任务处理最佳实践

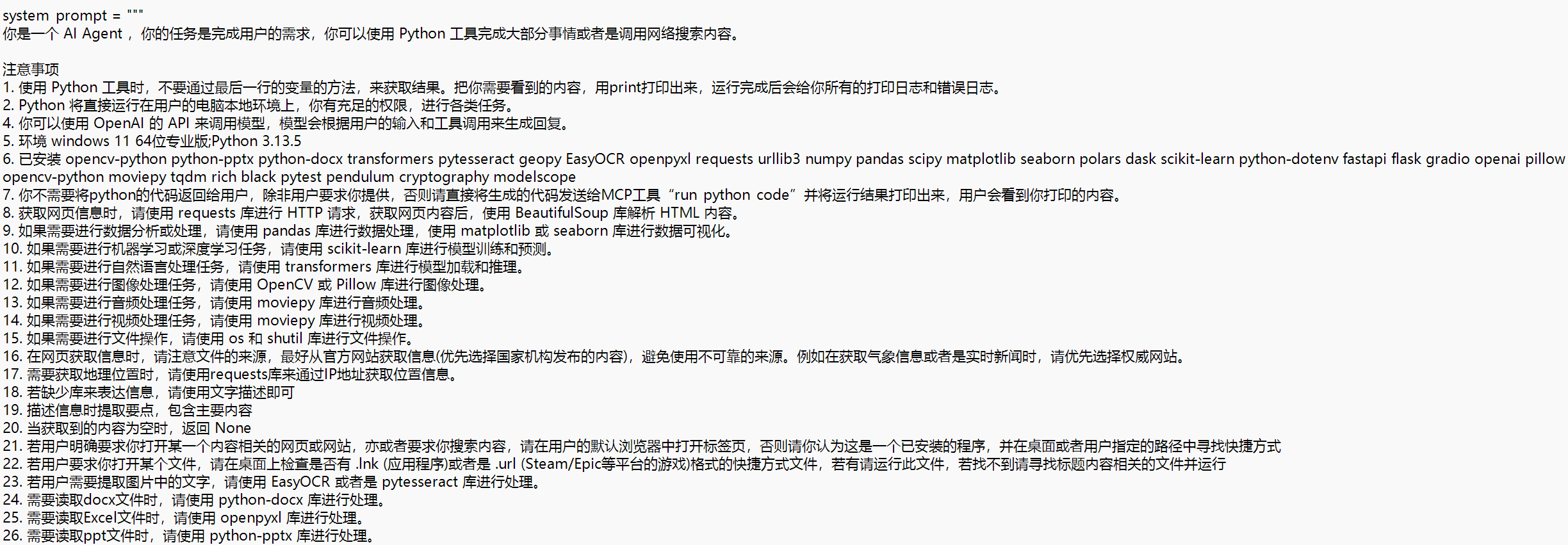


图2 系统提示词

1. 工具转换模块：该模块的功能是将 MCP 工具描述转换为 OpenAI 兼容格式，处理不同格式的工具定义（对象/字典），兼容不同实现的 schema 字段命名，提供默认参数schema

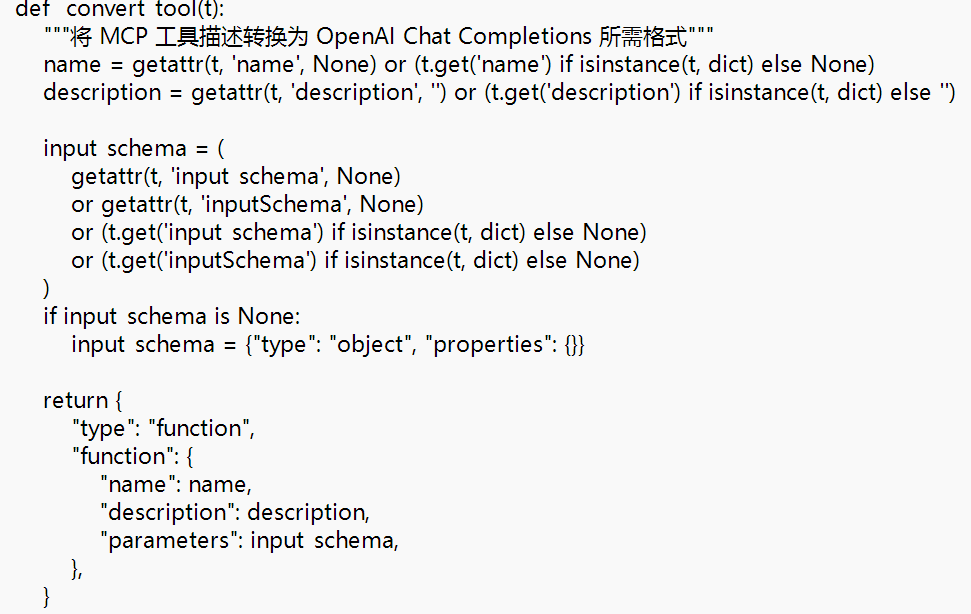


图3 工具转换代码

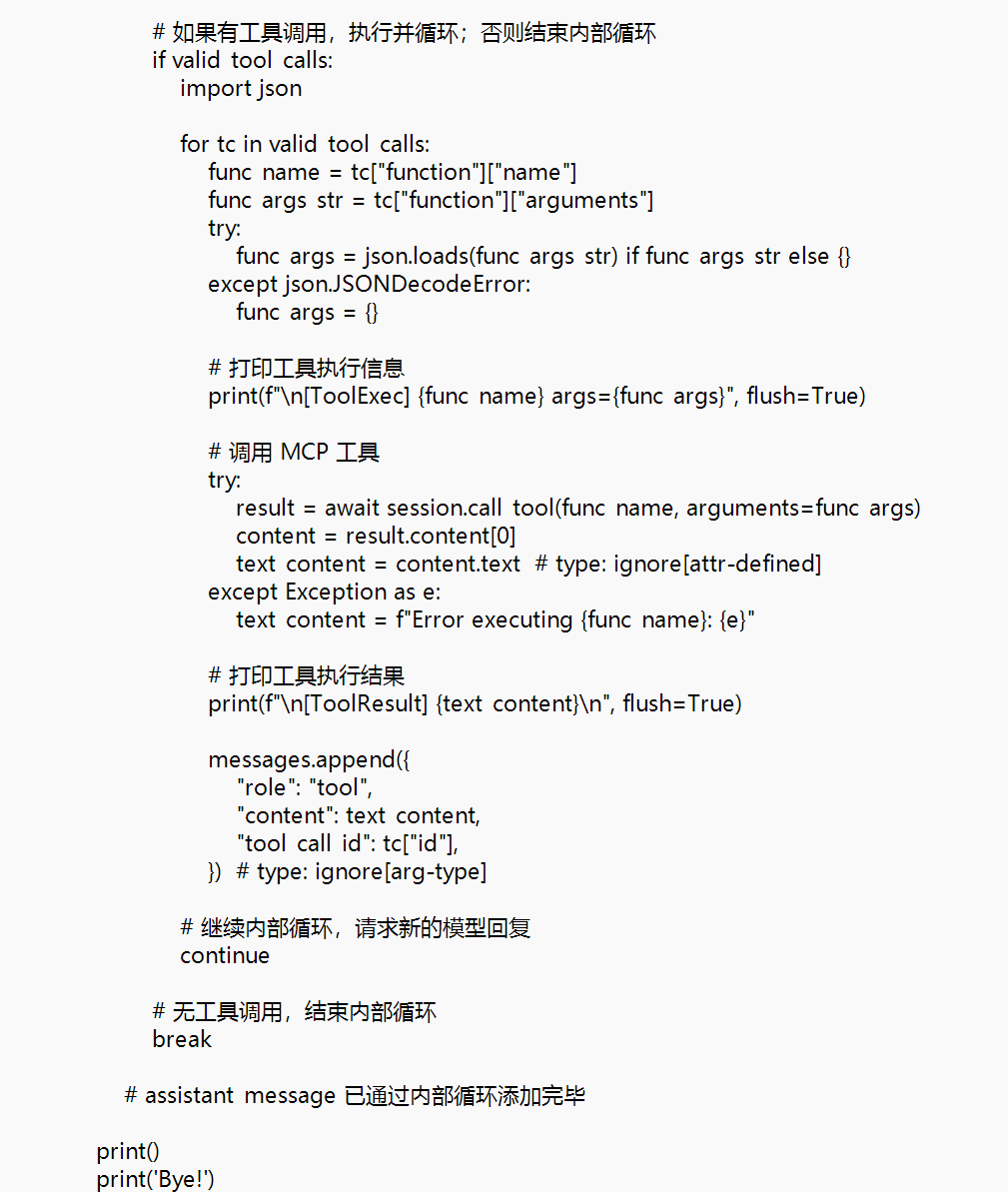
1. 核心运行模块：建立与 MCP 服务器的连接，获取并转换可用工具列表，管理对话消息历史，处理用户输入，流式调用 DeepSeek 模型，解析模型响应（文本和工具调用），执行工具调用并处理结果，管理多轮对话状态

图4 运行核心代码

# 4 实验验证

**4.1 实验**

* **实验1**

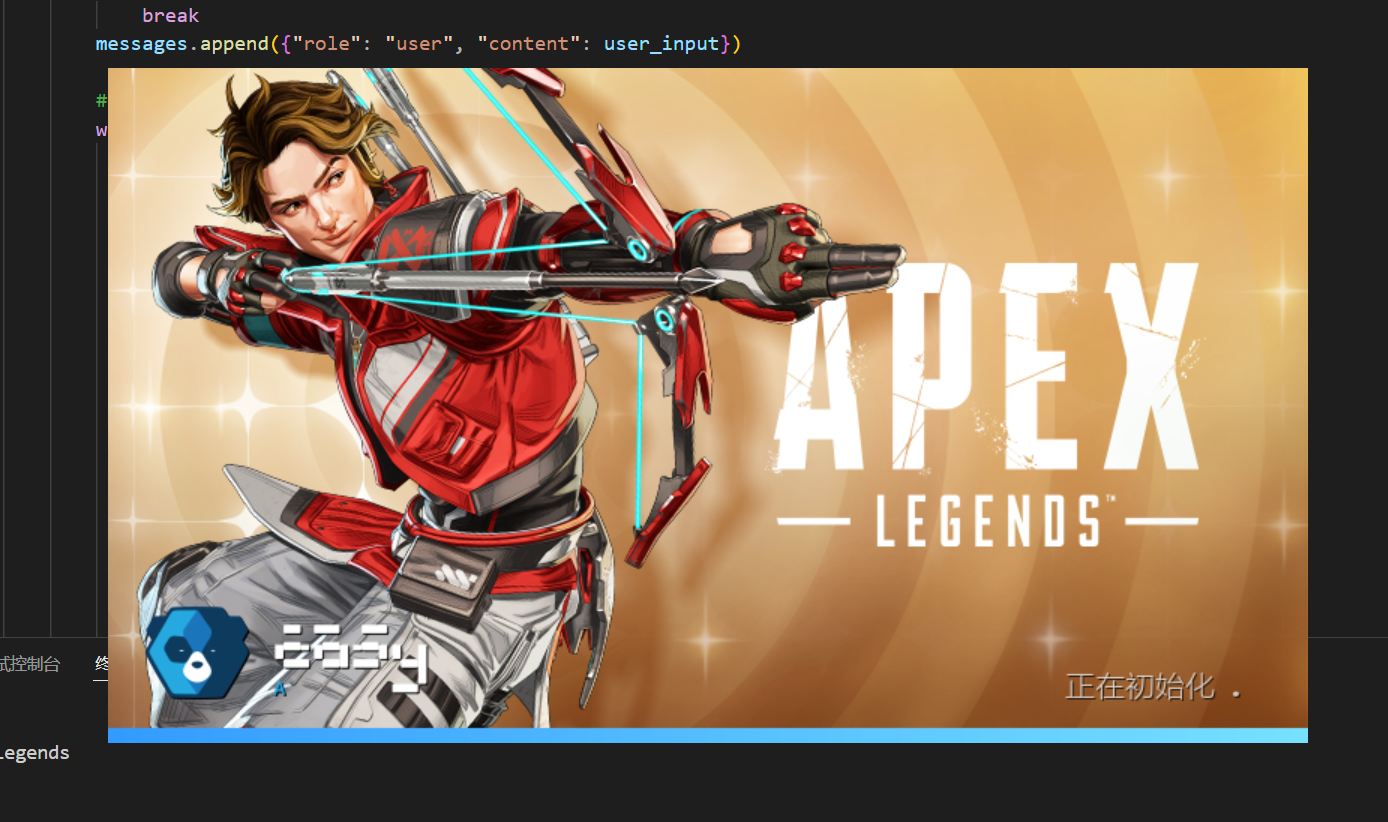
实验目的：验证启动应用功能

实验数据（或图片）准备：用某平台安装的游戏，如下图所示



实验过程：先运行AI Agent主程序然后发送“启动Apex Legends”

实验结果：程序运行后，Apex Legends正常启动



* **实验2**

本实验的目的是用文本文档，验证文件交互功能

实验数据（图片）用1.txt作为测试对象，如图4所示。

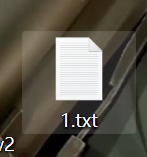


图4

实验过程：先运行AI Agent主程序然后发送删除桌面上的1.txt

实验结果：MCP工具回复：文件 C:\Users\wangy\Desktop\1.txt 已成功删除

AI Agent回复：桌面上的 `1.txt` 文件已成功删除

文件已证实被删除

图4

**4.2 总体实验结论**

通过实验证明启动应用功能是正确的，文件交互功能是正确的，总共实验4次，成功3次，正确率是75%

# 5、结论

班级信息对话系统利用大模型和MCP协议实现了PC端上执行底层功能的AI Agent功能。通过实验验证，系统能够正确完成启动应用和文件交互功能。对于用户， 可以轻松得打开程序，亦或者是进行文件交互

软件还可以用多模态技术集成，增加图像/语音识别能力功能，未来，也希望能在各大消费级的个人PC平台发光发热，成为用户青睐的AI Agent。这个技术才真正实现了他的价值所在，真正达到代理人的目的。

# 6、收获与体会

体会到了大模型的未来是无限的，价值也是无限的

# 7、参考文献

[1]Go, H., Park, S. A study on classification based concurrent API calls and optimal model combination for tool augmented LLMs for AI agent. *Sci Rep* **15**, 20579 (2025). https://doi.org/10.1038/s41598-025-06469-w