AI协作编程能力报告

本报告对测试者在四个编程任务中与AI协作的能力进行评估

题目一

错误修复与单元测试

2/2

题目二

LLM代理服务开发

1/10

题目三

舆情监控系统开发

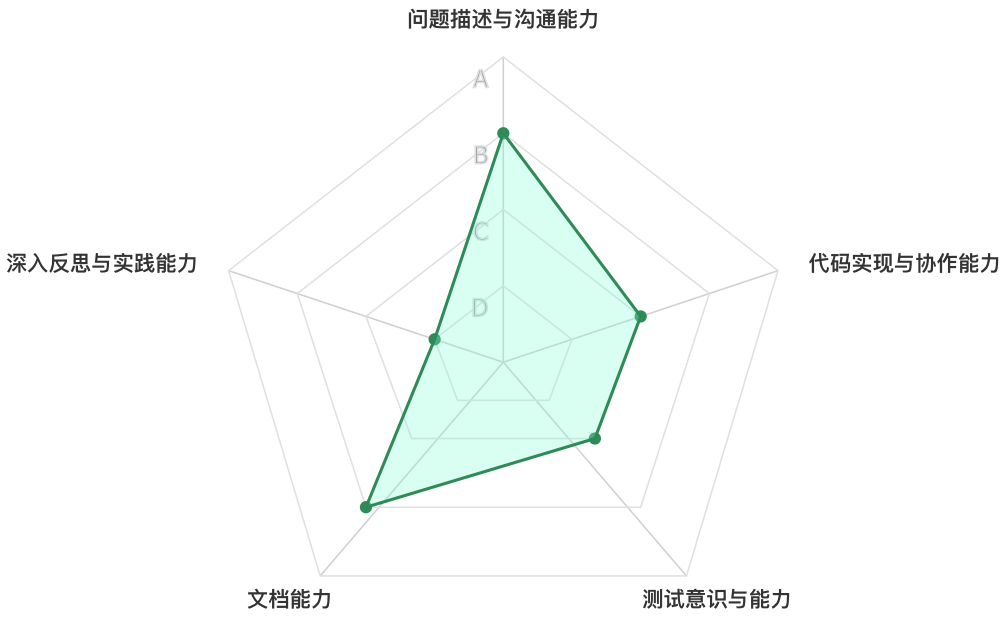
5/10

题目四

单元测试开发

7/10

综合能力评估



问题描述与沟通能力 (B)

准确有效地向AI描述需求和问题的能力。

用户能够在部分情况下清晰描述问题，尤其在热身题和AI辅助测试中，提供了明确的错误修复需求和测试要求。然而，在LLM Proxy和Agent开发中，用户的沟通缺乏深度，主要是重复任务描述，而没有提供有效的细化指导。例如，在调试失败后，用户没有主动分析错误，而是单纯要求AI"继续修改"，导致修复效率低下。这表明用户的沟通能力尚可，但在精确引导AI方面仍然较弱。

代码实现与协作能力 (C)

通过AI协作实现和优化代码的能力。

用户能够正确理解AI提供的代码，并在热身题中做出有效的修复。然而，在复杂任务中（如LLM Proxy和Agent开发），用户基本上依赖AI完成实现，而不是主动优化代码逻辑。例如，面对失败的测试，用户缺乏细化的debug思维，仅让AI自行摸索调整，而不是基于已有反馈指导AI修正代码结构。这表明用户具备一定的AI协作能力，但在深入理解和调整代码方面仍然存在较大提升空间。

测试意识与能力 (C)

设计有效测试用例并正确理解测试原理的能力。

用户在AI辅助测试和热身题中，展现了一定的测试意识，能够要求AI生成单元测试。然而，在Agent开发和LLM Proxy任务中，用户没有确保测试的全面性，尤其是在异常处理和边界条件测试方面。例如，用户在发现异常测试失败后，选择删除相关测试项，而不是修复代码中的错误。这种行为反映出用户的测试思维不够严谨，仍然停留在表层测试，而非系统性测试设计。

文档能力 (B)

撰写清晰易懂、实用性强的文档，辅助他人理解和使用的能力。

用户在AI辅助测试中有较好的文档意识，要求AI输出Markdown格式的bug分析。然而，在LLM Proxy和Agent开发中，用户没有提供足够的Prompt设计文档（PromptDoc）或架构说明（Design Doc），导致代码的可读性和可维护性较差。整体而言，用户的文档能力尚可，但缺乏在复杂项目中的深入应用，尤其是在涉及多人协作或长周期维护的任务中。

深入反思与实践能力 (D)

能够积极反思并推动AI进行更深入实践和优化的能力。

用户在整个过程中，几乎没有要求AI进行反思（Reflect），也没有主动检查AI代码的合理性。例如，在测试失败时，用户并未深入分析根因，而是选择删除测试项，以规避错误而非修正代码。此外，在LLM Proxy和Agent开发中，用户没有推动AI进行更深层次的优化，而是被动地让AI反复尝试修复。这表明用户的反思能力较弱，主要停留在表层修改，而缺乏系统性的思考和优化能力。

题目一：错误修复与单元测试 2/2

评估用户发言的有效性

用户的第一次发言有效：

- 用户明确引用了文件（`main.ts`）并要求找出错误。
- 助手正确指出并修复了原代码中的错误（处理空数组情况）。

用户的第二次发言也有效：

- 用户明确提出了生成单元测试的需求，要求考虑边界情况并给出了参考文件（`test.ts`）。
- 助手响应恰当，给出了全面的单元测试，包括空数组、负数、单元素、零、大数字等情况，测试用例全面且有效。

用户行为评分（满分2分）

错误修复 (1分)：

- 用户明确指出了`main.ts`中存在的问题（未处理空数组输入）。
- 用户接受并采纳了AI提出的修复建议（将`sum`初始化为0），有效解决了初始代码中存在的关键问题。
- 用户的修复行为准确、清晰，且能够有效避免程序运行时可能出现的异常情况。
- 得分：满分（1分）

单元测试撰写（1分）：

- 用户明确要求AI提供额外的单元测试，并具体强调了边界情况的考虑（如空数组、负数等）。
- AI按照用户的需求提供了丰富且完整的测试用例集合，包括空数组、单元素数组、负数数组、混合数组、零数组和大数字数组等各种情况。
- 用户虽未亲自撰写单元测试，但明确且具体地指示了AI进行这一操作，且AI生成的测试覆盖充分，满足了题目对单元测试的要求。
- 得分：满分（1分）

综合得分：

修复错误：1分

撰写单元测试：1分

总分：2 / 2 分

题目二：LLM代理服务开发 1/10

评估用户发言的有效性

用户的第一次发言：

> *README.md*
> *# LLM 代理服务器实现挑战*
> *[任务描述...]*

用户清晰地转述了README.md的任务内容，明确了本次实现的目标和要求。这是一个有效的起始请求。

后续发言：

- 用户在随后的发言中多次重复任务原文和代码文件内容，未提出有助于推动问题解决的具体问题或指导，导致AI缺乏有效的方向来逐步解决问题。
- 用户不断重复"参考 `@llmproxy/app.py` "，未能给AI提供进一步的信息或具体的修复方向，导致AI自行摸索代码修改和错误排查。
- 在单元测试失败后，用户给出了一系列的错误信息，但仍然未能提出明确的指导或具体的问题以帮助AI更高效地修复问题，只是泛泛要求AI"继续修改"。

用户行为评分（满分10分）

成功完整实现README的要求以通过测试（4分）：

- 用户进行了多次尝试，给出了明确的测试错误输出，但直到最后仍未能完全通过单元测试。
- 用户实现的代理功能本身相对简单且属于典型的AI zero-shot能力范围，未体现出额外的难度或特别的努力。
- 得分：1分（未成功完全通过测试，仅实现了部分基础功能，且属于zero-shot即可完成的任务）

能正确根据AI的回复去指导AI下一步的代码工作（4分）：

- 用户的多次发言内容几乎完全重复（例如反复粘贴原始任务要求），没有给出明确有效的指示，也未帮助AI识别具体的问题或修复方向。
- 用户未能有效利用AI给出的建议来提出明确的下一步行动，造成AI不得不多次自行摸索修改方向。
- 得分：0分（仅重复已有内容，无有效指导）

良好实践（2分）：

用户粉丝判断实现为代码逻辑而非prompt调用：

- 用户未做代码逻辑实现，而是始终调用 `gpt` 函数完成。
- 得分：0分 / 0.5分

是否实现累计舆情的判断（如过去24小时内出现多条负面舆情）：

- 用户给出的示例仅限单条情境演示，没有体现批量累计判断或流式处理的机制。
- 得分：0分 / 0.5分

综合得分：

实现要求及通过测试：1分

指导AI下一步代码工作：0分

良好实践及反思：0分

总分：1 / 10 分

题目三：舆情监控系统开发 5/10

评估用户发言的有效性

用户的发言过程如下：

- **第一次发言：**

  > *根据 @README.md 中提到的要求，参考 @main.py @test.py 生成agent代码和相应的单元测试*

  有效但比较泛泛，用户没有明确指出实现的具体细节或问题，仅让AI根据要求生成代码。

- **第二次发言：**

  > *根据 @README.md 中提到的要求，参考 @main.py 补充gpt函数内容，并提供相应的单元测试*

  用户提出了具体要求，即补充`gpt`函数并编写对应的测试，较第一次更明确，更有助于任务实现。

- **第三次发言：**

  > *根据 @README.md 中提到的要求，参考 @main.py @test.py 生成agent代码和相应的单元测试，保证参数正确引入，可正确运行*

  与第一次发言基本重复，没有实质性的新要求或反馈。这种重复提出类似请求的行为效率不高。

用户行为评分（满分10分）

函数输入与prompt自定义（4分）：

- 每个函数能够正确输入，且prompt构建在每个函数中
- prompt直接硬编码在函数内，缺乏灵活性
- 得分：3.5分

格式输出（1分）：

- 使用文本parse而非JSON，方法脆弱且易错
- 得分：0.5分

提示词文档设计（1分）：

- 未提供PromptDoc的相关说明
- 未明确提及输出格式文档
- 得分：0分

实现串联功能（1分）：

- 在main.py中清晰展示了多个步骤的串联
- 实现了完整的流程调用
- 得分：1分

设计决策说明（1分）：

- 未提供DesignDoc
- 未提及任何设计决策的记录
- 得分：0分

良好实践（2分）：

正确根据AI回复指导下一步的代码工作（1分）：

- 未展现对AI代码实现方向的检查或修正
- 得分：0分

引导AI进行reflect反思（1分）：

- 未要求AI反思或重新检查自身实现
- 得分：0分

额外得分（1分）：

代码逻辑实现vs prompt调用（0.5分）：

- 仅使用gpt函数调用，未做代码逻辑实现
- 得分：0分

累计舆情判断实现（0.5分）：

- 仅限单条情境演示，未实现批量累计判断
- 得分：0分

综合得分：

函数输入与prompt自定义：**3.5**分

格式输出：**0.5**分

提示词文档设计：**0**分

实现串联：**1**分

设计决策说明：**0**分

良好实践：**0**分

额外得分：**0**分

总分：**5 / 10** 分

题目四：单元测试开发 7/10

评估用户发言的有效性

用户的发言过程如下：

- **第一次发言：**

  > *对 @main.py 中的`group_items`函数编写单元测试。*
  > *识别并解释您的测试发现的任何潜在bug或边界情况。*
  > *简要解释为什么在您的解决方案中包含每个单元测试。*

  这是一个有效的请求，明确指出了测试目标和要求。

- **第二次发言：**

  > *将对bug和边界情况的分析以md格式返回*

  体现了良好的文档化意识。

- **第三次发言：**

  > *单元测试文件中少了引入：*
  > *from main import group_items*

  展现了代码审查能力。

- **第四次发言：**

  > *FFF.F.F.F（单元测试失败信息），根据执行的错误信息，更正单元测试*

  提供了详细的测试失败信息。

- **最后的发言：**

  > *@test.py 中，去掉 ValueError 和 TypeError 相关的单元测试项*

  反映了错误的测试实践。

用户行为评分（满分10分）

测试案例覆盖（4分）：

初始测试用例包含：
    - 空输入列表
    - 列表长度能整除分组大小
    - 列表长度不能整除分组大小
    - 分组大小为1
    - 分组大小大于列表长度
    - 分组大小为0或负数（后被删除）
    - items为None（后被删除）

- 虽然初始覆盖全面，但后续删除关键异常测试项导致覆盖不完整
- 得分：3分

测试解释和边界情况分析（2分）：

生成的分析文档包含：
    - 详细的输入情况描述
    - 预期行为分析
    - 实际观察结果

- 得分：2分

测试意识（2分）：

存在明显问题：
    - 发现异常处理问题时未推动主函数修复
    - 选择删除测试项而非解决根本问题

- 得分：0分

良好实践（2分）：

表现出的良好实践：
    - 要求以Markdown格式输出分析结果
    - 主动提供错误信息用于问题定位
    - 保持良好的调试和反馈习惯

- 得分：2分

综合得分：

测试案例覆盖：**3**分

测试解释和边界情况分析：**2**分

测试意识：**0**分

良好实践：**2**分

总分：**7 / 10** 分