

DDL 战神计划

核心原则：不懂就问 AI，代码跑通为主，不要追求完美。

🛠 Day 0: 全员准备

1. 编辑器: VS Code (必装插件: Python, Pylance)。
2. AI 助手: 每个人必须装 Cursor 或 GitHub Copilot, 或者网页版常驻 ChatGPT/Claude/DeepSeek。
3. 环境: 统一使用 `Python 3.12+`, 使用虚拟环境。
4. 代码同步: 推荐用 GitHub Desktop (可视化点一点就能提交, 防止命令行敲错)。

👨‍💻 P1: 队长 & 基建工 (Team Lead & Infra) []

你的角色：管线铺设者。大家都要用你写的函数来调 AI。你的工具：`LangChain` (或者直接用 `openai` 库),

📅 Day 1: 环境与接口

- 任务: 初始化 Git 仓库, 创建 `.gitignore` (把 `.env` 和 `venv` 加上)。
- 任务: 编写核心文件 `app/core/llm_client.py`, 支持切换七牛API接入。
- 任务: 创建 `.env.example` 模板文件, 教会队友怎么填 API Key。
- 交付物: 一个能运行的 `main.py`, 打印出 "Hello from DeepSeek"。

📅 Day 2: 串联与重试

- 任务: 写一个 `load_prompt(yaml_path, variables)` 函数, 帮 P2 读取 YAML 文件并填入变量。
- 任务: 给 `call_llm` 加上简单的重试机制。如果报错, 自动重试 1 次 (用 `tenacity` 库, 问 AI 怎么写)。
- 交付物: 健壮的 `llm_client.py`, 队友调用时不容易崩。

📅 Day 3: 兜底与部署

- 任务: 增加全局异常捕获。如果 LLM 返回的 JSON 格式烂了, 你的函数要 catch 住报错, 返回一个默认的错误提示, 别让整个后端挂掉。
- 任务: 确保演示时, 本地 `uvicorn run` 能一次跑通。
- 交付物: 最终演示版本代码。

⚠ 新手注意：千万别把真实的 API Key 提交到 GitHub 上！一定要检查 `.env` 是否被忽略了。

P2: 提示词工程师 (Prompt Engineer) 【韩明烨】

你的角色：教 AI 说话的人。你不太需要写 Python 逻辑，但要懂怎么指挥 AI。你的工具：VS Code (写 YAML), AI Playground (在线测试)。

Day 1: 提取与拆解

- 任务: 在项目根目录建 `app/prompts/` 文件夹。
- 任务: 为 4 个功能各写一个 YAML 文件 (如 `generate_tests.yaml`)。把代码里写死的 Prompt 搬进去。
- 任务: 确定 Prompt 里的挖坑变量 (例如 `{source_code}`, `{diff_content}`)，发给 P3 确认。canvas
- 交付物: 4 个基础 Prompt 文件。

Day 2: 举例与优化 (Few-Shot)

- 任务: 找 P5 要几个“标准答案”。
- 任务: 在 Prompt 里加入 `Examples:` 章节。告诉 AI “输入A应该是输出B”，这能极大提高准确率。
- 任务: 针对 P4 反馈的问题 (比如 AI 喜欢说废话)，在 Prompt 里加“严禁废话，只返回代码”。
- 交付物: v2 版本的高质量 Prompt。

Day 3: 压力测试

- 任务: 故意塞一段很烂、很长的代码给 Prompt，看 AI 会不会胡言乱语。
- 任务: 如果 AI 崩了，在 Prompt 开头加防御指令：“如果代码无法解析，请返回空 JSON”。
- 交付物: 最终版 Prompt 包 + 简单的使用说明文档。

 新手注意：YAML 文件里的变量名 (如 `{code}`) 必须和 P3 代码里传的一模一样，差一个字母都会报错。

P3: 业务逻辑开发 (Logic Developer) 【施可婳】

你的角色：搬砖主力。把 P1 的接口、P2 的 Prompt 和 P3 的数据串起来。你的工具：`Python ast` 库, `GitPython` 库 (直接问 AI 怎么用)。

Day 1: 解析代码

- 任务: 问 AI “写一个 Python 函数，利用 AST 库提取代码里的所有函数名和类名”。
- 任务: 问 AI “如何用 Python 解析 git diff 字符串，获取变更的行号”。
- 交付物: `app/analysis/ast_parser.py` 和 `diff_parser.py`。

17 Day 2: 混合分析逻辑

- 任务 (Impact Analysis): 写逻辑流程 -> 解析 Git Diff -> 找到变更函数 -> 调用 P1 的接口 -> 拿到结果。
- 任务 (Quality Analysis): 用 Python 的 `subprocess` 模块调用 `pylint` 命令，拿到报错文本，只把报错的部分喂给 LLM。
- 交付物: `app/routers/impact.py` 和 `quality.py`。

17 Day 3: 清洗与修复

- 任务: LLM 经常返回 ````json` 这种 Markdown 格式。写个正则（问 AI）把这三个点删掉，只提取纯 JSON。
 - 任务: 联合 P1 测试，确保前端传过来的参数（如 `source_code`）你能正确读到。
 - 交付物: 能跑通的业务 API。
- ⚠ 新手注意 :** JSON 解析是最容易报错的地方。遇到 `JSONDecodeError` 不要慌，把 AI 返回的字符串打印出来看一看，通常是多了个逗号或者引号没闭合。

17 P4: 评测与验证 (Metrics & Testing) 【常俊然】

你的角色：裁判。你要证明我们的 AI 是有用的。你的工具：`pytest`, `subprocess`。

17 Day 1: 搭建评测脚本

- 任务: 写一个 `evaluate.py` 脚本。
- 逻辑: 读取生成的测试代码 -> 写入一个临时的 `.py` 文件 -> 运行 `pytest 临时文件.py` -> 捕获是成功还是失败。
- 交付物: 一个能自动跑测试的 Python 脚本。

17 Day 2: 批量跑分

- 任务: 用 P5 准备的数据，通过 P1 的接口跑 10-20 次。
- 任务: 记录数据：DeepSeek 生成的代码，10 个里有几个能跑通？几个报错？
- 反馈: 拿着报错的例子去找 P2，“你看，它老是忘记 import pytest，快去改 Prompt”。
- 交付物: 初步的评测 Excel 表格。

17 Day 3: 最终报告

- 任务: 跑完所有数据。
- 任务: 算几个简单的指标：编译通过率 (Compilability) 和 运行通过率 (Pass Rate)。
- 交付物: `最终评测报告.md` (哪怕只提升了 10% 也是提升，诚实记录即可)。

P5: 数据专家 (Data Specialist) 【骆彦伶】

你的角色：矿工。没有你的数据，P4 就没法测，P2 也没法调优。你的工具：GitHub, Excel/JSONL 编辑器。

17 Day 1: 考古挖掘

- 任务: 去 GitHub 搜简单的 Python 项目（推荐 `requests`, `flask` 的早期版本, 或者包含 `calculator` 的简单练习项目）。
- 任务: 找到带有 `test` 或 `fix` 关键词的提交 (Commit)。
- 任务: 保存“修改前的代码”作为 Input。
- 交付物: `data/raw/` 文件夹下的原始代码文件。

17 Day 2: 构造答案 (Ground Truth)

- 任务: 那个提交里, 开发者手写的测试代码, 就是“标准答案”。保存下来。
- 任务: 清洗数据。确保你找的代码是独立的（不需要连数据库、不需要复杂的配置就能跑）。如果太复杂, 手动删掉一些无关代码。
- 交付物: `data/dataset.jsonl` (整理好的测试集)。

17 Day 3: 找茬 (Edge Cases)

- 任务: 专门造几个“坏”数据：空文件、全是注释的文件、只有一行代码的文件。
- 任务: 给 P3 测试, 看系统会不会崩。
- 协助: 帮 P4 整理图表数据。
- 交付物: `data/edge_cases.json`。

 新手注意：贪多嚼不烂。找 10 个简单清晰的小文件, 比找 100 个复杂的大文件要有用得多！

Tips

1. 遇到报错怎么办？

- 复制报错的最后一行 (Error Message)。
- 粘贴到 Cursor/AI 里, 问：“我这段代码报这个错, 是什么原因, 怎么改?”
- 不要自己死磕超过 15 分钟。

2. Git 冲突怎么办？

- 如果 GitHub Desktop 提示冲突, 不要慌。
- 把代码拉下来, 手动保留大家都有的部分。
- 如果搞不定, 就把自己的文件重命名 (比如 `main_p3.py`) 先传上去, 让 P1 来合并。

3. AI 也会骗人

- AI 可能会调用一个不存在的 Python 库。如果你 `pip install` 找不到，那说明 AI 在瞎编。让它换一个标准的库。