

AI 全流程开发实战

从一句需求到线上验收

Claude Code + WSL + Figma MCP + Chrome MCP + 编译型后端

不是提示词大全，而是一条可落地、可验收的工程流水线。

10

阶段

7

门禁

3 周

MVP

95%

bug 可提前发现



中国区，怎么用上最强的 Agent ？

客户端和模型解耦：Claude Code CLI + DeepSeek V4 Flash ；接 DeepSeek 必须跳过权限确认。

Claude Code CLI

目前最领先的 CLI Agent （按当前实践）

MCP · 子 Agent · hooks · skills · 全流程工作流

DeepSeek V4 Flash

后端模型：能力 + 速度

按当前路由，Pro 的流量也走 Flash ； Flash 是 DeepSeek 当前最强

生成速度（示意）

DeepSeek V4 Flash

快

GLM / Kimi

相对慢

体感：Flash 明显更快；能力也足够打。

接 DeepSeek 必须：**`claude --dangerously-skip-permissions`**

否则每个子命令都要确认

客户端选 Claude Code ，模型选 DeepSeek ：工程能力 + 生成速度兼得。

前提：项目在 Git 里；所有改动经 git diff / 文件审查后再合并；生产环境不使用 bypass 。

先问三个问题

不用回答我，先在心里举手。

01

AI：写代码还是交付？

代码生成很快，但需求、契约、验收、上线仍然散落。

02

你说“完成”时，有证据吗？

单测？E2E？截图？Word 报告？还是“AI 说可以了”？

03

你敢让 AI 直接上线吗？

没有门禁、回滚、监控，越快越危险。

答案不是“别用 AI”，而是“给 AI 一套工程流程”。

60 分钟，讲清一条可落地的流水线

每个部分都有“能立刻用”的产物。

00-17 开场 + 心智模型 中国区模型 / 三种模式 / 产能公式 / 上下文 / 门禁	17-28 需求与设计 spec / 用户故事 / CSV / 契约冻结	28-36 多 Agent 编码 任务书 / worktree / TDD / 报告	36-48 质量与发布 Chrome MCP / Word / CI/CD / 线上	48-60 落地与 Q&A 30 分钟闭环 / 反模式 / 行动清单
---	---	---	---	---

1 一张图 10 阶段 / 7 门禁，知道每一步该做什么	2 一套产物 spec / CSV / OpenAPI / 报告 / Word
3 一套规则 多 Agent 并行：worktree + 文件所有权	4 一个闭环 30 分钟跑通一个用户故事

01

心智模型

AI 是工程团队，不是许愿池。

先改变你看 AI 的方式，再谈工具。



三种工作模式，风险完全不同

大多数人停在 Copilot ； Team 模式需要流程。

Copilot

副驾驶

改一个函数、写一个脚本、解释代码

人的参与

人逐步确认

风险：低

Pair

结对

一个模块、一次重构、一个页面

人的参与

关键节点确认

风险：中

Team

工程团队

完整产品、多个功能并行

人的参与

门禁确认

风险：高，需要流程

本培训讲第三种： Team 。

AI 工程产能 = 六个因子的乘积

不是加法，是乘法：任何一项接近 0，整体产能就接近 0。



模型能力 架构 / 审查用强模型，机械任务用快模型	上下文质量 文件 > 聊天；一故事一会话	任务分解 小步、可验收、可并行
反馈速度 编译 / 测试 / Chrome MCP 快速证伪	验证强度 单测 + 集成 + E2E + 人	人的判断 目标、边界、审美、上线

短板决定产能：先把最弱的一项拉起来。

把 AI 拆成一支工程团队

每个角色都有输入、输出、权限和完成标准。

产品负责人（你）

1 需求

spec / 故事 / 验收

2 架构

契约 / ADR / 边界

3 UI

设计稿 / token

4 前端

页面 / 状态 / 联调

5 后端

API / DB / 测试

6 测试

E2E / 截图 / Word

7 发布

CI/CD / 冒烟 / 回滚

8 审查

安全 / 契约 / 测试

人的时间花在“判断”，不是“搬运”。

聊天会忘，文件不会

不要让 AI “记住”，要让 AI “读文件”。

聊天记录



- 不可检索：三天后找不到“当时为什么这么定”
- 不可并行：多个 Agent 无法可靠共享上下文
- 不可审计：出了问题无法回溯假设和决策

文件系统

```
project/  
  
CLAUDE.md  
docs/00-discovery/  spec / CSV / details  
docs/01-design/     OpenAPI / schema / ADR  
docs/02-ui/         ui-spec / screens  
docs/03-dev/reports/ 一个故事一份报告  
docs/04-qa/         manifest / 截图 / Word
```

每个阶段的结论，都必须落成仓库里的文件。

上下文是稀缺资源

上下文越干净，AI 越聪明，token 越省。



- 一个会话只做一个故事
- @ 文件精确引用，不让 AI 自己找
- rg 定位，大文件只读片段
- 子 Agent 做搜索和日志摘要
- 做完 /clear ，重开读契约

七个门禁：把返工成本往前推

门禁不是审批，是止损。



越晚发现，修复成本越高



每个门禁只问一个关键问题（示例）

- G1：现在停下来，损失最小吗？
- G4：如果两个 Agent 现在并行，会冲突吗？
- G6：如果我是用户，我敢用吗？
- G7：如果上线失败，我能在 10 分钟内退回吗？

02

需求与设计

先冻结契约，再并行开发。

需求不清，AI 会非常勤奋地编。



需求阶段的四件套

不是写一份长文档，而是让每个 Agent 能回答：做什么、不做什么、怎么算做完。

1

spec.md

目标 / 非目标 / FR / NFR / 权限 / 异常

2

user-story.csv

唯一状态源：优先级、状态、负责人、链接

3

user-story-details.md

业务规则 / 边界 / API / 数据 / 验收

4

schedule.md

里程碑 / 依赖 / 关键路径 / 缓冲

关键：每个用户故事都必须有可测试的验收标准。

用户故事不是一句话，是一份可验收契约

故事负责价值，验收标准负责边界。

故事格式

作为 <角色>，
我希望 <能力>，
以便 <价值>。

示例：US-004 创建任务

作为 项目成员，
我希望 创建一个带负责人和截止日期的任务，
以便 我能把工作分配清楚。

验收标准（Given / When / Then）

AC-1 正常创建 201 + 卡片出现

AC-2 标题为空 400 + 字段错误

AC-3 无权限 403 + 提示

AC-4 日期非法 400 + 提示

验收标准的质量，决定 E2E 的质量。

CSV 是唯一状态源

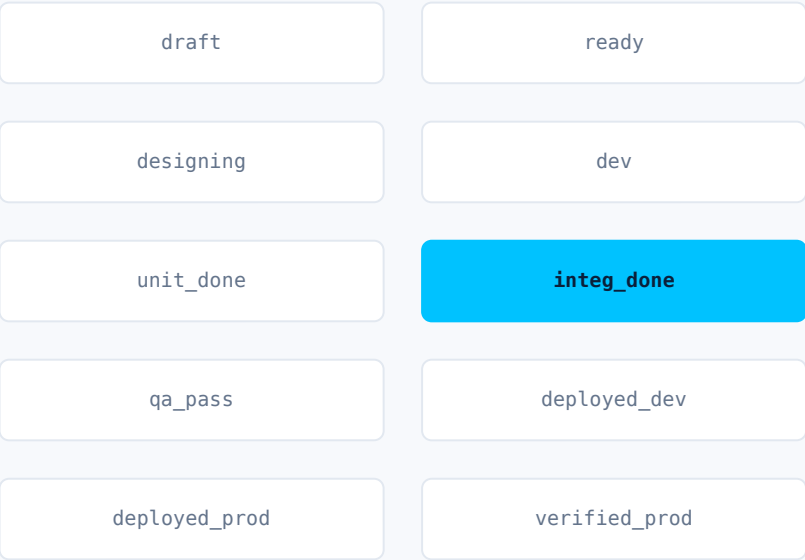
状态只能由脚本更新，不能手改 CSV。

story_id	title	priority	status	owner
US-001	用户登录	P0	qa_pass	backend-1
US-002	创建项目	P0	qa_pass	backend-2
US-003	邀请成员	P0	integ_done	backend-2
US-004	创建任务	P0	integ_done	backend-3
US-005	任务看板	P1	dev	frontend-1

● ● ● update_story_status.py

```
python3 scripts/update_story_status.py \
--file docs/00-discovery/user-story.csv \
--story US-004 --status integ_done \
--report docs/03-dev/reports/US-004.md
```

状态机（按生命周期流转）



CSV 还索引：改动文件、功能报告、 E2E manifest 、各环境部署状态。

技术设计：三件套 + 一张施工图

契约优先：先写接口和表，再写实现。

tech-design.md

架构 / 组件 / 数据流 / 并发 / 安全 / 部署

openapi.yaml

前后端唯一接口事实：字段、错误、分页

schema.sql

表 / 索引 / 约束 / 迁移 / 回滚

feature-implementation.md

每个故事怎么落地：事务、错误码、测试、文件

OpenAPI + schema 是机器可读的契约；feature-implementation 是 Agent 的施工图。

G4 契约冻结：并行的前提

契约不冻结，并行就是制造冲突。

1

API 契约

字段 / 错误码 / 分页 / 幂等

2

DB 模型

表 / 索引 / 约束 / 迁移顺序

3

模块边界

adapter → usecase → domain

4

文件所有权

谁能改哪些文件；共享文件串行

冻结后才能开多个 Agent；任何变更走 CR，不悄悄改契约。

请求隔离：多 Agent 并行的底线

请求之间互不干扰，Agent 之间才不会互相踩踏。

无全局可变状态

依赖注入，运行期只读

每请求 context

超时、request_id、user/tenant

每个写请求一个事务

任务 + 审计 + outbox 同事务

幂等 + 连接池上限

Idempotency-Key；MaxConns

Go：每请求 context + 超时

```
ctx, cancel := context.WithTimeout(r.Context(), 5*time.Second)
defer cancel()

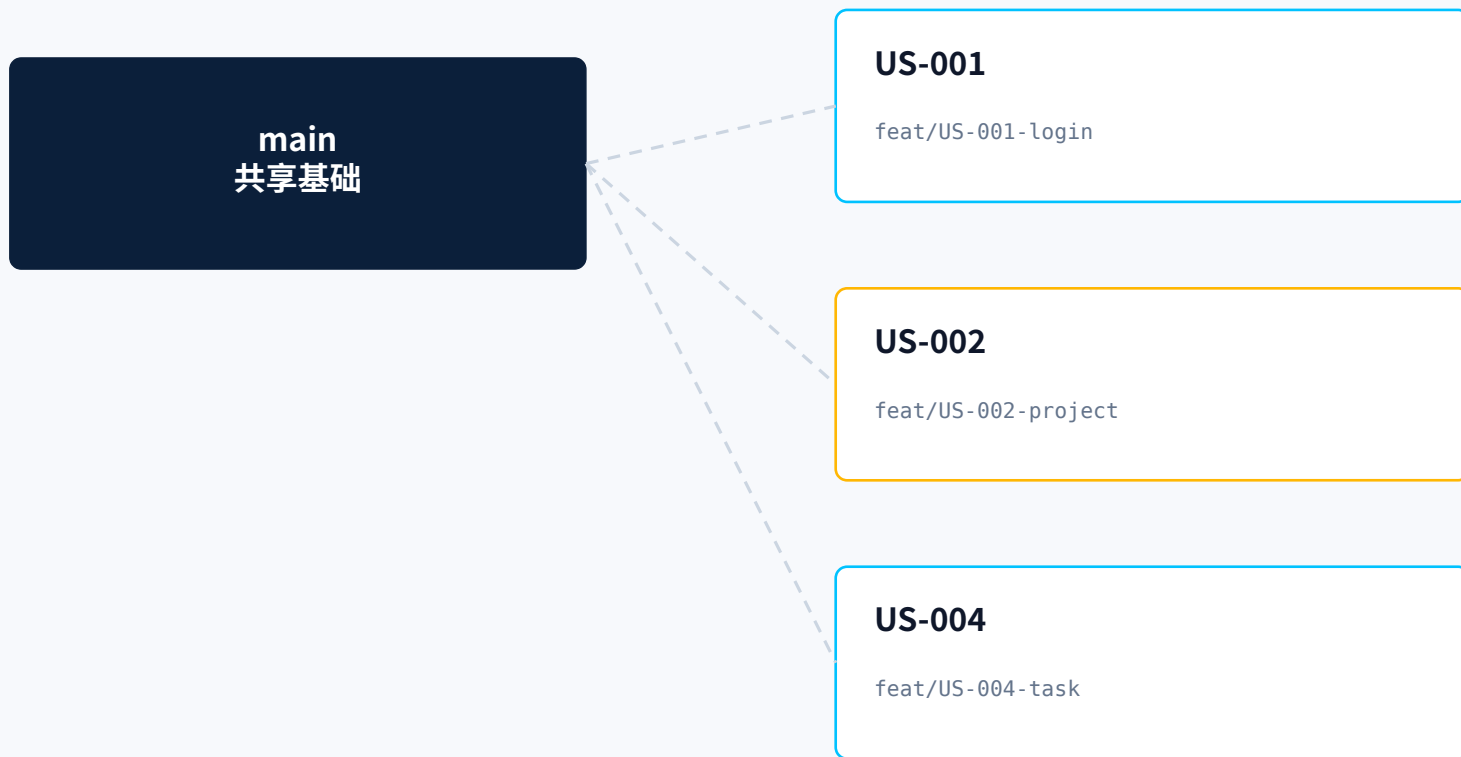
// 用户 / 租户只从 context 取，不存全局
user, ok := auth.UserFrom(ctx)
tenantID := auth.TenantIDFrom(ctx)

// 写请求：任务 + 审计 + outbox 在同一事务
err := s.db.WithTx(ctx, func(tx pgx.Tx) error { ... })
```

AI 很容易写全局变量；这是架构评审的必查项。

一个 Agent 一个 worktree，一个文件只有一个主人

并行度：第一次 2 个 Agent，熟练后 3-4 个。



并行交通规则

- 共享文件只由架构 Agent 改
- 迁移文件统一编号、串行
- 冲突时停止，不自动 ours/theirs
- 一个故事一个会话

并行度：2-3 个 Agent 起步

```
git worktree add ../worktrees/US-004 -b feat/US-004-task main
```

CLAUDE.md：项目宪法

每次会话自动加载；短而关键，细节放 docs/。

应该写什么

- 一句话：产品为谁解决什么问题
- 技术栈、目录、常用命令
- 编码约定：错误码、UTC、权限在后端
- 禁止事项：不改未授权文件、不删测试、不泄密
- 完成定义：测试、报告、CSV、审查

只放长期约定；临时状态放 CSV。

CLAUDE.md (节选)

```
## 常用命令
make dev / make test / make lint
make e2e / make report

## 禁止
- 不改 router.go / main.go / migrations
- 不删 / 跳过测试让 CI 变绿
- 不把密钥写进代码或日志

## 完成定义
- 单测 + 集成测试全绿
- 生成功能报告并更新 CSV
- 代码审查无 P0/P1
```

控制在 250 行以内；它是宪法，不是百科全书。

03

多 Agent 编码

让多个 Agent 像一支工程队。

任务书 + *worktree* + *TDD* + 报告。



谁干什么：一张 Agent 分工表

角色清晰、权限清晰、模型档位清晰。

Agent	职责	并发	模型
架构	共享文件 / 路由 / 迁移 / 契约变更	串行	最强
后端功能	一个故事的后端 + 单测 + 集成	并行	中 / 强
前端功能	页面 / 组件 / 联调 + 测试	并行	中 / 强
集成	合并 / 全量测试 / 冲突处理	串行	强
测试	E2E / 截图 / Word 报告	并行	中
审查	只读：安全 / 契约 / 测试	并行	最强
发布	构建 / 部署 / 冒烟 / 回滚	串行	中

审查 Agent 只读；发布 Agent 执行但不批准；人保留最终批准权。

任务书：Agent 的开工合同

没有停止条件的 Agent，会为了“完成”而改契约、删测试。

1	目标	交付什么，做到什么程度
2	边界	可改 / 禁改文件，共享文件走 CR
3	契约	API / 错误码 / 事务 / 幂等
4	测试	单测 / 集成 / 命令 / 覆盖率
5	停止条件	契约冲突 / 需要共享文件 / 测试不明

```
agent-brief.md (节选)

# Agent 任务书：US-004 创建任务

## 文件所有权
可改: internal/usecase/task/**
禁止: router.go / main.go / migrations

## 契约
POST /api/v1/tasks
400 / 403 / 404 / 409

## 完成定义
- go test -race ./... 全绿
- 生成功能报告 + 更新 CSV

## 停止条件
契约冲突 / 需要共享文件 / 测试不明
```

停止条件是防跑偏的关键：契约冲突 / 共享文件 / 测试不明 → 停下来报告。

TDD 循环：让测试先失败

先看到红色，才能证明测试真的在测东西。

- 1

读契约

OpenAPI / schema / 实现细节
- 2

写失败测试

空标题 / 无权限 / 冲突
- 3

最小实现

只让测试通过
- 4

重构

保持测试绿
- 5

lint / 集成

race / vet / lint / 全量

TDD 循环

```
go test ./internal/usecase/task/...  
FAIL TestCreateTask_EmptyTitle (red)  
  
// 写最小实现  
func (s *Service) Create(...) (*Task, error) { ... }  
  
go test -race ./...  
PASS ok internal/usecase/task (green)  
  
// 补边界：无权限 / 项目归档 / 幂等
```

测试质量三问：会失败吗？覆盖错误码 / 权限吗？有并发测试吗？

AI 很容易写“永远通过”的测试；要求先红后绿，并审查测试质量。

一个故事，一份报告，一行状态

各功能不共用报告；CSV 是索引，不是日记。

功能报告（一个故事一份）

- 基本信息：故事 / 分支 / Agent / 时间
- 改动文件：逐文件说明
- 实现要点：权限、事务、并发、错误
- 测试结果：单测 / 集成 / race / lint
- AC 覆盖：每条验收标准对应的测试
- 已知限制：没做完的、有风险的
- 证据：commit / CI / manifest / 截图

字段	值
story_id	US-004
status	integ_done
feature_report	docs/03-dev/reports/US-004.md
e2e_manifest	docs/04-qa/manifests/US-004.json
dev_deploy	deployed

●●● 更新状态

```
python3 scripts/update_story_status.py \  
  --story US-004 --status integ_done \  
  --report docs/03-dev/reports/US-004.md
```

合并顺序，决定并行成败

并行不是目的，减少返工才是。

推荐合并顺序

1	共享基础	main / router / migrations
2	认证	US-001 登录
3	项目 / 成员	US-002 / US-003
4	任务	US-004 / US-005
5	前端	按页面依赖
6	测试修复	全量 + E2E

上下文与成本

- 一个故事一个会话，做完 /clear
- @ 文件精确引用，不让 AI 自己找
- 机械任务用快模型；架构 / 审查用强模型
- 截图只存文件，不回读模型
- 冲突时停止，不自动解决

38 亿

tokens / 月

¥171

常规开发成本

经验值：一个故事 30-90 分钟；人工审查 15-30 分钟。

04

质量与发布

没有证据的完成，不算完成。

编译器 + 测试 + E2E + 人。



四层防线，层层过滤

编译器是测谎仪，不是业务验收员。



编译型语言解决“低级幻觉”；业务幻觉靠测试、契约和验收。

AI 生成太快，人的注意力跟不上

解法不是“多审查”，而是“让 AI 生产证据，人只做判断”。

矛盾

AI 生成速度

↑ 10×

人的注意力

→ 1×

代码越多，审查越堵；靠人肉看不过来。

产能瓶颈从“写”转移到了“看”。

用 AI 治理 AI

- 1 AI 生成全流程文件 spec / 契约 / 任务书
- 2 AI 生成测试 单测 / 集成 / E2E
- 3 AI 自动执行 CI / Chrome MCP / 截图
- 4 AI 生成人性化报告 Word / 结论 / 缺陷

人只做三件事：定目标、过门禁、做最终判断。

Chrome MCP：用文本 + JS 思考，用截图留证

图片回读模型太贵；文本和 JS 更准、更省。

工具	用途
chrome_navigate	打开 URL
chrome_read_page	可访问性树 / 元素
chrome_javascript	执行 JS，返回 JSON
chrome_console	控制台错误
chrome_network_capture	网络 / 状态码
chrome_screenshot	截图留证（只存文件）

●●● 用 JS 断言，不靠看图猜

```
// 页面状态
({
  path: location.pathname,
  hasCard: !!document.querySelector(
    '[data-testid="task-card"]')
})

// 表单错误
document.querySelector(
  '[data-testid="title-error"]')
?.textContent?.trim()
```

截图：storeBase64 = false，保存到磁盘，嵌入 Word，不回读模型。

能用一行 JS 拿到的事实，不要用截图。

每一步截图，最后变成一份 Word

人像审文档一样审流程，不用来回点浏览器。

1

执行步骤

navigate / click / fill

2

JS 断言

预期 vs 实际

3

截图保存

storeBase64=false

4

写 manifest

步骤 / 断言 / 错误

5

生成 Word

python-docx

manifest.json (节选)

```
{ "story_id": "US-004", "result": "pass",  
  "steps": [  
    { "index": 1, "action": "navigate", "status": "pass",  
      "screenshot": "screens/US-004/step-01.png" },  
    { "index": 5, "action": "create-task", "status": "pass",  
      "assertions": [{ "expected": "201", "actual": "201" }] } ] }
```

每一步都有：动作、预期、实际、截图、结果。

Word 报告 + 人工走查

人负责判断，AI 负责修改和取证。

报告结构

- 封面：故事 / 环境 / 时间 / 结果
- 概要：通过 / 失败 / 缺陷统计
- 验收标准覆盖：AC → 步骤 → 结果
- 测试步骤：动作 / 预期 / 实际 / 截图
- 控制台与网络：错误 / request_id
- 缺陷清单：严重度 / 状态 / 修复
- 证据文件：manifest / 截图 / CI 日志
- 结论与签字：验收人 / 日期 / 遗留问题

Word = 可读证据 + 可签字结论

人工走查流程

- | | | |
|---|---------|-----------|
| 1 | 打开 Word | 先看封面结论 |
| 2 | 逐页看截图 | 对照预期 / 实际 |
| 3 | 批注问题 | 在 Word 里标 |
| 4 | 让 AI 修 | 附失败证据 |
| 5 | 重跑失败故事 | 生成新报告 |

人只做 AI 不擅长的判断：交互、文案、视觉、异常体验。

部署： AI 可以执行，人必须批准

生产发布 = 受控、可回滚、可验证的状态变更。



AWS 目标

- 前端： S3 + CloudFront
- 后端： ECS Fargate / Lambda
- 数据库： RDS + 迁移任务
- 回滚： 任务定义 / 镜像版本
- 回滚顺序： 前端 → 后端 → 数据库

Syncthing 同步 AWS 登录态

- 同步： ~/.aws/config + sso/cache
- 不要同步： ~/.aws/credentials
- 用 SSO 短期凭证，最小权限
- 凭证过期就重新登录，不自动绕过

线上验证：真实环境再走一遍

不通过就回滚或关开关，不能“先这样上线”。

1

专用测试租户

数据前缀 e2e-`<run>-<story>`-

2

只读优先

写操作只碰自己创建的测试数据

3

Chrome MCP

每一步截图 + manifest

4

Word + 监控

线上报告 + 错误率 / 延迟基线

线上验证失败：先回滚 / 关功能开关，再定位和修复。

05

落地

从一个最小闭环开始。

不要一次改造整个团队。



30 分钟最小闭环

先跑通一个故事，再扩到 3 个 Agent。

1 选故事

最小、可演示

5 min

2 写 spec + AC

正常 + 2 异常

5 min

3 冻结契约

OpenAPI + schema

5 min

4 实现 + 单测

一个 Agent

8 min

5 E2E + 截图

Chrome MCP

4 min

6 Word + CSV

报告 + 状态

3 min

课后作业：用你自己的一个小功能，跑一遍这 6 步。

六个最容易翻车的地方

每个反模式，都有一个门禁或模板对应。



没有验收标准就开工

解药：G1 + Given/When/Then



契约没冻结就并行

解药：G4 + OpenAPI + schema



多个 Agent 改同一文件

解药：worktree + 文件所有权



删测试让 CI 变绿

解药：零容忍 + 审查测试



截图全读回模型

解药：文本 /JS 优先，截图只存文件



生产测试用真实数据

解药：测试租户 + 数据前缀

明天就能做的七件事

先做小，再做快；先做对，再做多。

- ☒ 选一个 3-5 个故事的小项目
- ☒ 写 spec + user-story.csv + details
- ☒ 冻结 OpenAPI + schema + 文件所有权
- ☒ 建 CLAUDE.md + Agent 任务书
- ☒ 用 worktree 开 2 个 Agent
- ☒ 跑 Chrome MCP E2E + Word 报告
- ☒ 人工走查并签字

Q&A

- 你的项目最该先补哪道门禁？
- 你准备先让哪个角色 Agent 化？
- 你的第一个 30 分钟闭环是什么？
- 哪个反模式你踩过？

课后作业：用一个小功能跑通 6 步闭环

资料：Word 教程 · 12 模板 · 5 脚本

TAKEAWAY

先冻结契约，再并行开发。
没有证据的完成，不算完成。

谢谢 · 欢迎交流