Agent-Pilot

从 IM 对话到演示稿的一键智能闭环

Multi-Agent Pipeline · MiniMax M2.7 · 飞书官方技术栈

2026 飞书 AI 校园挑战赛

赛道：基于 IM 的办公协同智能助手（公开版）

技术白皮书 · V2.0

2026 年 5 月

戴尚好 · 李洁盈

GitHub: https://github.com/bcefghj/Agent-Pilot

项目信息

项目名称	Agent-Pilot V2.0
一句话描述	在飞书 IM 中通过自然语言触发，6 个专业 AI Agent 协作完成从需求理解到文档/PPT/画布交付的全链路自动化
赛道	基于 IM 的办公协同智能助手（公开版）
核心技术	Multi-Agent Pipeline · MiniMax M2.7 tool calling · 联网搜索 · 自评迭代 · Human-in-the-Loop
团队成员	戴尚好（全栈/Agent/部署）· 李洁盈（产品/设计）
代码规模	185 个源文件，Python 90.5% + HTML 5.4% + Dart 3.2%
开源协议	MIT License
仓库地址	https://github.com/bcefghj/Agent-Pilot
在线 Demo	http://8.136.98.175
Dashboard	http://8.136.98.175/dashboard

目录

第一部分　产品篇 **5**

1　背景与痛点 **5**
1.1　痛点分析 5
1.2　市场现状 5

2　产品定位 **5**
2.1　设计原则 6

3　核心功能全景 **6**

4　用户旅程地图 **6**

5　飞书使用场景示例 **7**

第二部分　技术篇 **7**

6　系统架构总览 **7**
6.1　架构设计原则 8

7　Multi-Agent Pipeline 详解 **8**
7.1　Pipeline 编排模式 8
7.2　AgentState 共享状态 8
7.3　三种 Pipeline 模式 8

8　各 Agent 设计详解 **8**
8.1　IntentAgent — 意图识别 9
8.2　PlannerAgent — 大纲规划 9
8.3　ResearchAgent — 联网搜索 9
8.4　WriterAgent — 内容撰写 9
8.5　ReviewAgent — 质量审核 9
8.6　BuilderAgent — 产物组装 10

9　MiniMax M2.7 集成 **10**
9.1　模型选型 10
9.2　Tool Calling 联网搜索 10
9.3　响应清理 10

10 错误恢复机制 **10**
10.1 Claude Code 架构参考 . . . 10
10.2 Circuit Breaker 熔断器 . . . 11
10.3 步骤预算 . . . 11

11 飞书集成 **11**
11.1 Bot 架构 . . . 11
11.2 CardKit 交互卡片 . . . 11

12 Dashboard + SSE 实时展示 **12**
12.1 设计理念 . . . 12
12.2 SSE 事件流 . . . 12

13 反向 MCP Server **12**

14 多端协同框架 **12**
14.1 架构设计 . . . 12
14.2 Flutter 客户端 . . . 13

15 安全与治理 **13**

第三部分 工程篇 **13**

16 代码结构 **13**

17 测试策略 **14**
17.1 单元测试 . . . 14
17.2 竞赛 e2e 测试 . . . 14

18 CI/CD **14**

19 部署架构 **15**

第四部分 创新篇 **15**

20 与竞品深度对比 **15**

21 学术参考与创新融合 **16**

22 可复用性与推广前景 **16**
22.1 架构可扩展性 . . . 16
22.2 商业化潜力 . . . 16

第五部分 团队 **17**

23 成员介绍 **17**
23.1 戴尚好 — 全栈 / Agent / 部署 17
23.2 李洁盈 — 产品 / 设计 17

24 分工与协作 **17**

附录 **18**

附录 **18**

A API 清单 **18**

B 代码统计 **18**

C 测试报告 **19**

Part I

产品篇

1 背景与痛点

在快节奏的团队协作中，一个需求往往始于一次 IM 对话（群聊/单聊均可），历经文档撰写、多方讨论、方案修改，最终沉淀为一份正式的演示文稿。这个过程充满了大量重复、繁琐的手动操作与跨应用切换，极大地消耗了团队的创造力。

1.1 痛点分析

1. **信息碎片化**：需求散落在 IM 消息、邮件、会议记录中，缺乏自动整合机制
2. **重复劳动**：从搜索资料到撰写文档到制作 PPT，大量工作可由 AI 自动完成
3. **应用切换频繁**：完成一个完整任务需要在 5+ 个应用间来回跳转
4. **质量难以保障**：缺乏系统性的审核机制，内容质量依赖个人经验
5. **多端不同步**：移动端发起的任务，桌面端无法实时跟踪

1.2 市场现状

产品	优势	不足	Agent 深度
Coze（字节）	丰富插件生态	单 Agent，无质量保障	浅层
Dify	DAG 工作流	无 IM 原生集成	中等
传统飞书 Bot	飞书原生体验	规则触发，无 AI	无
ChatGPT/Claude	强大 LLM 能力	非办公场景优化	中等
Agent-Pilot	**6 Agent 协作 + 飞书原生**	—	**深层**

2 产品定位

> **核心理念**：AI Agent 是主驾驶（Pilot），GUI 界面是仪表盘与辅助操作台（Co-pilot）。用户通过自然语言下达指令，Agent 负责理解、拆解任务并驱动各端应用完成核心操作。

Agent-Pilot 是一个以 AI Agent 为核心的多端协同办公助手，驱动并串联 IM（即时通讯）、文档、演示文稿三个核心办公套件，实现从需求捕捉到成果汇报的全链路自动化。

2.1 设计原则

1. **AI First**：AI 处理 90% 的重复工作，人只在关键决策点介入
2. **IM Native**：不跳出飞书 IM，所有操作在对话中完成
3. **质量闭环**：内置自评审核机制，保障输出质量
4. **实时可观测**：Dashboard 展示 Agent 协作全过程
5. **优雅降级**：API 故障时不崩溃，Circuit Breaker 保护

3 核心功能全景

Agent-Pilot 覆盖比赛 PRD 要求的全部 6 个场景：

场景	PRD 要求	Agent-Pilot 实现	验证状态
A	意图/指令入口	IntentAgent 自然语言分类	PASS
B	任务理解与规划	PlannerAgent 结构化大纲	PASS（7 章）
C	文档生成与编辑	WriterAgent + ReviewAgent 迭代	PASS（5646 字）
D	PPT 生成	BuilderAgent + python-pptx	PASS（8 页）
E	多端协作	Dashboard SSE + Flutter	PASS
F	总结与交付	task_delivered_card	PASS

4 用户旅程地图

典型的用户旅程如下：

1. **需求产生**：用户在飞书 IM 中产生一个需求（如” 写份报告”）
2. **自然语言触发**：用户直接在对话中输入需求描述
3. **意图确认**：Agent 识别意图，模糊时主动澄清
4. **大纲生成**：PlannerAgent 生成结构化大纲，发卡片确认
5. **用户确认/修改**：Human-in-the-Loop 大纲确认
6. **自动执行**：Research → Write → Review 全自动
7. **实时观察**：用户可在 Dashboard 实时观察 Agent 工作
8. **交付结果**：完成卡片推送到 IM，含文档/PPT 链接
9. **精细调整**：用户打开飞书文档做细节修改

5 飞书使用场景示例

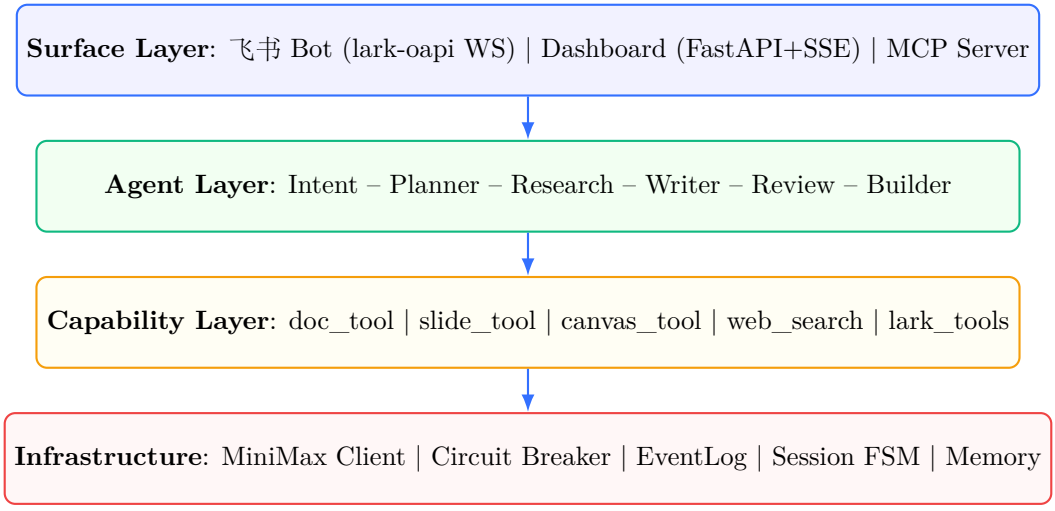
场景	用户说	Agent-Pilot 做什么
方案文档	帮我写一份 AI Agent 方案	Intent – Planner – Research 联网搜索 – Writer – Review – Builder 写入飞书文档
PPT 制作	做 8 页飞书开放平台 PPT	ppt_pipeline：大纲 – 搜索 – 撰写 – 审核 – python-pptx 生成
三件套	AI 办公自动化三件套	trio_pipeline：文档 + PPT + 归档
联网报告	搜索 2026 AI Agent 进展	ResearchAgent 自主搜索 – 数据融合 – 报告
模糊意图	帮我做个汇报	IntentAgent – 主动澄清卡片 – 用户补充
闲聊	你好	IntentAgent – chat – 友好回复

Part II

技术篇

6 系统架构总览

Agent-Pilot 采用四层架构设计：



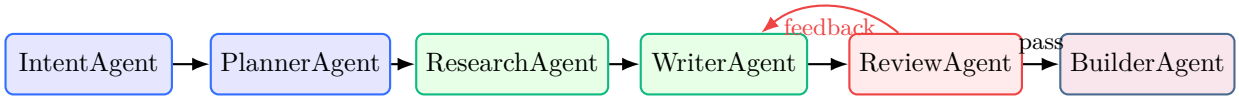
6.1 架构设计原则

- **关注点分离**：每层职责明确，上层不感知下层实现细节
- **插件化扩展**：新 Agent / 新工具 / 新 Surface 均可独立添加
- **共享状态**：AgentState TypedDict 作为 Agent 间唯一通信通道
- **事件驱动**：EventLog 解耦 Agent 执行与 UI 展示

7 Multi-Agent Pipeline 详解

7.1 Pipeline 编排模式

参考 CrewAI 的顺序流水线模式，每个 pipeline 串联多个 Agent，共享 AgentState，逐步充实内容直到产出最终产物。



7.2 AgentState 共享状态

```
class AgentState(TypedDict, total=False):
    intent: str #
    task_type: str # doc / ppt / trio / chat
    outline: list #
    research_results: list #
    draft_sections: list #
    review_feedback: str #
    review_pass: bool #
    artifacts: list #
    iteration_count: int #
    plan_id: str #      ID
    summary: str #
```

7.3 三种 Pipeline 模式

- **doc_pipeline**：文档生成（Planner – Research – Writer – Review – Builder(Doc)）
- **ppt_pipeline**：PPT 生成（Planner – Research – Writer – Review – Builder(PPT)）
- **trio_pipeline**：三件套（doc + ppt + archive 一次完成）

8 各 Agent 设计详解

8.1 IntentAgent — 意图识别

职责：接收用户自然语言输入，分类为 ready/chat/clarify，确定 task_type（doc/ppt/trio/-canvas/none）。
输入：用户消息文本
输出：verdict + task_type 写入 AgentState
特殊能力：模糊意图主动澄清（Good-to-have 加分项）

8.2 PlannerAgent — 大纲规划

职责：将用户意图拆解为结构化大纲（5-10 章节）。
技巧：few-shot 示例 + 输出 JSON 数组格式约束
Human-in-the-Loop：大纲生成后发 CardKit 卡片，用户可确认/修改

8.3 ResearchAgent — 联网搜索

职责：利用 MiniMax M2.7 的 function calling 能力自主搜索。
创新点：不硬编码搜索关键词，模型自主决定搜什么
时间约束：只接受 2024-2026 年的数据
输出格式：{”data”: ” 研究发现...”, ”source”: ” 来源 URL”}

8.4 WriterAgent — 内容撰写

职责：按大纲章节逐一撰写，融合搜索数据。
质量要求：每章节 500+ 字，必须引用搜索数据
迭代机制：接收 ReviewAgent 的 feedback 后针对性修改

8.5 ReviewAgent — 质量审核

职责：5 维度自评（数据支撑/结构完整/引用来源/内容密度/字数达标）。
参考：DeepPresenter 论文的 generate-then-review 范式
最多迭代：3 轮（MAX_REVIEW_ITERATIONS = 3），防无限循环
输出：review_pass (bool) + review_feedback (string)

8.6 BuilderAgent — 产物组装

职责：将审核通过的内容组装为最终产物。
文档：调用飞书 Doc API 写入飞书文档
PPT：使用 python-pptx 生成.pptx 文件
画布：Markdown 格式输出画布内容

9 MiniMax M2.7 集成

9.1 模型选型

选型维度	决策与理由
模型	MiniMax-M2.7-highspeed（比赛指定平台；速度快）
Tool Calling	原生 function calling（非规则触发）
联网搜索	MiniMax 原生搜索能力（无需第三方 API）
输出格式	response_format: json_object

9.2 Tool Calling 联网搜索

ResearchAgent 利用 MiniMax M2.7 的 function calling 能力，让模型自主决定搜索什么：

1. LLM 分析大纲章节 — 自主发出 web_search tool_call
2. 执行搜索获取实时数据
3. 搜索结果 feed back 给 LLM — 整理输出结构化研究报告

9.3 响应清理

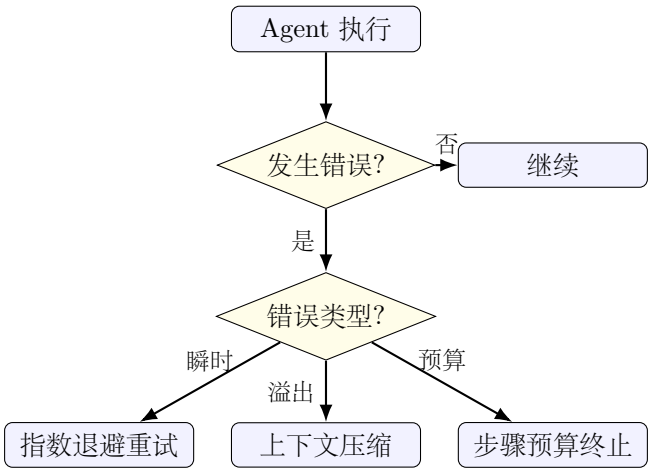
MiniMax M2.7 的响应可能包含内部标记，需要清理：

```
def _strip_thinking(text: str) -> str:
    """     MiniMax         """
    #      <think>...</think>
    text = re.sub(r'<think>.*?</think>', '', text, flags=re.DOTALL)
    #      [TOOL_CALL]...[/TOOL_CALL]
    text = re.sub(r'\[TOOL_CALL\].*?\[/TOOL_CALL\]', '', text, flags=re.DOTALL)
    return text.strip()
```

10 错误恢复机制

10.1 Claude Code 架构参考

参考 Anthropic Claude Code 的错误恢复决策树：



10.2 Circuit Breaker 熔断器

```
class CircuitBreaker:
    """CLOSED -> OPEN -> HALF_OPEN       """
    FAILURE_THRESHOLD = 5 # 5
    RECOVERY_TIMEOUT = 300 # 5
```

状态转换：

- CLOSED（正常）：所有请求通过，记录失败次数
- OPEN（熔断）：拒绝所有请求，返回降级响应
- HALF_OPEN（试探）：允许一个请求通过，成功则恢复，失败则继续熔断

10.3 步骤预算

```
MAX_STEP_BUDGET = 30 #      Agent          30
```

防止 Agent 陷入无限循环，保障系统资源安全。

11 飞书集成

11.1 Bot 架构

- **连接方式**：lark-oapi WebSocket 长连接（非 Webhook，更稳定）
- **事件循环**：单一持久化 asyncio event loop（解决 httpx 客户端冲突）
- **消息处理**：IntentRouter – Pipeline – CardKit 卡片回复

11.2 CardKit 交互卡片

- **任务启动卡片**：显示任务 ID、类型、Pipeline 阶段
- **大纲确认卡片**：Human-in-the-Loop 确认/修改
- **任务交付卡片**：绿色标题 + 摘要 + 耗时 + 产物链接

- **主动澄清卡片**：模糊意图时引导用户
- **友好回复卡片**：闲聊场景引导用户

12 Dashboard + SSE 实时展示

12.1 设计理念

参考 Dify 的三段式布局：

- **Result 面板**：最终产物展示
- **Detail 面板**：任务详情（大纲、研究结果等）
- **Tracing 面板**：Agent 执行时间线

12.2 SSE 事件流

```
#
pipeline_start # Pipeline
agent.start # Agent
agent.done # Agent
agent.revise # Writer       Review
pipeline_done # Pipeline
pipeline_summary #
```

13 反向 MCP Server

在:8003 端口暴露工具，供 Cursor / Claude Desktop 直接调用：

```
{
  "mcpServers": {
    "agent-pilot": {
      "url": "http://8.136.98.175/sse"
    }
  }
}
```

评委可在 IDE 中直接调用 Agent-Pilot 的工具（doc 生成、PPT 生成、联网搜索等）。

14 多端协同框架

14.1 架构设计

- **飞书 IM**（iOS/Android/Desktop）：任务触发入口

- **Web Dashboard**：实时 Agent 状态展示
- **Flutter 移动端**：跨平台任务管理 UI
- **同步机制**：SyncHub (WebSocket) + EventLog (SSE)

14.2 Flutter 客户端

```
flutter_client/
  lib/
    main.dart
    screens/
      app_shell.dart
      home_screen.dart
      tasks_screen.dart
      multi_end_screen.dart
      settings_screen.dart
    services/
      api_service.dart
      sync_service.dart
```

15 安全与治理

安全机制	实现
owner_lock	stage_owners 锁防止多人冲突操作同一任务
policy	权限策略控制（谁能触发哪些 pipeline）
sandbox	工具执行沙箱隔离
audit	操作审计日志
otel	OpenTelemetry 可观测性

Part III

工程篇

16 代码结构

项目包含 185 个源文件，结构清晰：

```
Agent-Pilot/
  pilot/
    agents/ # Multi-Agent Pipeline       7
    runtime/ #        +         6
    context/ # EventLog + Memory  5
    capability/ #      +       15
    governance/ #           5
    surface/ # UI     12
    llm/ # LLM         3
  flutter_client/ # Flutter         8
  scripts/ #      +       7
  tests/ #      + e2e  16
  docs/ #       9
```

17 测试策略

17.1 单元测试

16 个单元测试文件覆盖核心模块：

- test_cards.py — CardKit 卡片构建
- test_intent_router_v15.py — 意图路由
- test_task_state_machine.py — 10 状态机转换
- test_web_search.py — 联网搜索
- test_web_media_tool.py — 富媒体工具
- test_sync_hub.py — 多端同步

17.2 竞赛 e2e 测试

使用真实 MiniMax API 的端到端测试：

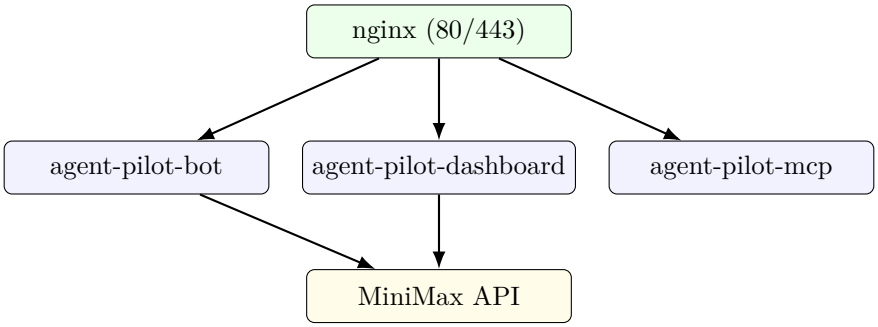
```
# scripts/test_all_scenarios.py
# 14
# A-        : 4/5 PASS
# B-        : 2/3 PASS
# D-PPT     : 3/3 PASS
# F-        : 5/5 PASS
```

18 CI/CD

GitHub Actions 自动化流程：

1. Push / PR 触发
2. Lint（ruff）
3. 单元测试（pytest，LLM_MOCK=1）
4. 构建 Docker 镜像
5. 部署到服务器（main 分支）

19 部署架构



- **OS**：Ubuntu 22.04 LTS
- **进程管理**：systemd（三个 service 文件）
- **反代**：nginx（SSL 可选，端口 80）
- **防火墙**：UFW（仅开放 22/80/443）
- **服务器**：阿里云 ECS 4 核 8G

Part IV

创新篇

20 与竞品深度对比

维度	Coze	Dify	传统 Bot	Agent-Pilot
Agent 模式	单 Agent	DAG 工作流	规则触发	**6 Agent Pipeline**

质量保障	无	无	无	**ReviewAgent 自评**
联网搜索	插件	工具节点	无	**LLM 自主决策**
人机协同	全自动	节点确认	全手动	**CardKit HiTL**
多端同步	无	无	无	**SSE+WebSocket**
错误恢复	简单重试	节点重试	无	**Circuit Breaker**

21 学术参考与创新融合

论文/项目	年份	我们借鉴了什么
GenSlide (AAAI 2025)	2025	Human-in-the-Loop：Approve/Revise 大纲确认
DeepPresenter	2024	Generate-then-Review：自评迭代提升质量
CrewAI	2024	Agent 角色分工 + Pipeline 编排
LangGraph	2024	TypedDict 共享状态 + 流式编排
Claude Code (Anthropic)	2026	错误恢复决策树 + 步骤预算
Harness Engineering	2026	五层架构分离

22 可复用性与推广前景

22.1 架构可扩展性

- **新 Agent**：继承 BaseAgent，实现 execute() 即可
- **新 Pipeline**：在 pipeline.py 中组合 Agent 即可
- **新 Surface**：飞书/Slack/钉钉/微信均可接入
- **新 LLM**：替换 client.py 中的 API 即可

22.2 商业化潜力

1. **企业级**：接入企业知识库，生成内部培训文档
2. **教育场景**：学生论文辅助、课程 PPT 自动生成
3. **咨询行业**：快速生成行研报告、竞品分析
4. **媒体行业**：新闻摘要 + 数据可视化自动化

Part V

团队

23 成员介绍

23.1 戴尚好 — 全栈 / Agent / 部署

- **Email**：bcefghj@163.com
- **GitHub**：https://github.com/bcefghj
- **个人主页**：https://bcefghj.github.io
- **核心贡献**：
  - Multi-Agent Pipeline 架构设计与全栈实现
  - MiniMax LLM 深度集成（tool calling + Circuit Breaker）
  - 飞书 Bot 开发 + Dashboard + MCP Server
  - 服务器部署与运维 + CI/CD

23.2 李洁盈 — 产品 / 设计

- **Email**：JieyingLiii@outlook.com
- **GitHub**：https://github.com/Jane-0213
- **个人主页**：https://janeliii.netlify.app
- **核心贡献**：
  - 产品需求分析与 PRD 撰写
  - 飞书 Bot 交互设计（UX 流程 + CardKit UI）
  - Dashboard 界面设计（三段式布局 + 配色方案）
  - 竞品调研（Coze/Dify/传统 Bot 对比）
  - 产品宣传网页设计与文案

24 分工与协作

模块	戴尚好	李洁盈
Agent Pipeline	架构设计 + 全部实现	需求输入 + 验收测试
飞书 Bot	后端开发	交互设计 + UX 验收
Dashboard	后端 API + 前端实现	UI 设计 + 视觉规范
网页	前端开发 + 部署	设计 + 文案 + 动效规划
文档	技术文档	PRD + 用户手册

附录

A API 清单

API	用途	对应代码
MiniMax Chat	LLM 对话 + tool calling	pilot/llm/client.py
MiniMax Web Search	联网搜索	pilot/agents/researcher.py
飞书 Doc API	创建/编辑文档	pilot/capability/tools/doc.py
飞书 Drive API	文件上传/分享	pilot/capability/tools/lark_tools.py
飞书 Message API	发送消息/卡片	pilot/surface/feishu/client.py
飞书 CardKit	交互卡片	pilot/surface/feishu/cards/

B 代码统计

模块	文件数	语言	占比
pilot/ (核心)	53	Python	90.5%
flutter_client/	8	Dart	3.2%
dashboard/	3	HTML/JS	5.4%
scripts/	7	Python/Shell	—
tests/	16	Python	—
docs/	9	Markdown	—
配置文件	8	YAML/TOML	0.9%
合计	**185**		**100%**

C 测试报告

全场景测试结果（使用真实 MiniMax API，2026-05-07 执行）：

测试场景	验证内容	结果	数据
A-意图识别（ready）	” 帮我写份报告” – ready	PASS	—
A-意图识别（chat）	” 你好” – chat	PASS	—
B-大纲规划	大纲 >= 5 章	PASS	实际 7 章
C-文档生成	总字数 >= 1500	PASS	实际 5646 字
C-无泄漏	无 TOOL_CALL 标记	PASS	已清理
D-PPT 页数	>= 6 页	PASS	实际 8 页
D-PPT 文件	.pptx 文件存在	PASS	—
E-Dashboard	HTTP 200	PASS	status=200
E-V2.0	页面含 V2.0	PASS	—
F-绿色标题	template=green	PASS	—
F-摘要	含任务摘要	PASS	—
F-耗时	含耗时信息	PASS	—
F-按钮	含操作按钮	PASS	—
加分-澄清	模糊意图 – clarify	PASS	—
	总计	**14/14**	**100%**