📋 OmniBriefing: 基于 MCP 的智能简报系统开发计划

版本: 1.2 (Added: MCP Resources, Memory & Cost Opt)

状态: 🟢 Sprint 1 准备中

架构模式: Client-Server (MCP Protocol with Resources & State)

🌟 项目愿景 (Project Vision)

构建一个具备**长期记忆**和**自我配置能力**的自主智能体。它通过 MCP 接口搜集天气、金融及新闻数据，能够根据用户对话动态调整偏好，并利用历史记忆生成具有连续性的、低成本高价值的每日简报。

🛠 技术栈 (Tech Stack)

- **核心语言**: Python 3.10+
- **Agent 协议**: MCP (Tools, Resources, Prompts)
- **大模型 (LLM)**:
  - **Tier 1 (重逻辑)**: Google Gemini 1.5 Pro (用于最终生成、复杂冲突分析)
  - **Tier 2 (轻量级)**: Google Gemini 1.5 Flash (用于 Gatekeeper、分类、提取关键词) —— **核心策略：利用 Free Tier**
- **数据源**: OpenWeatherMap, yfinance, Serper/Brave
- **状态管理**: JSON / SQLite (用于存储历史记录和偏好)

🏃 Sprint 1: The Hello World (Weather Agent)

主题: 搭建 MCP 基础架构，跑通最小闭环。

ID	User Story (用户故事)	Acceptance Criteria (验收标准)
US-01	作为用户，我希望获得今日天气提醒。	1. Agent 能获取指定城市天气。 2. 输出为自然语言建议。
US-02	作为开发者，我希望系统基于 MCP 架构。	1. 成功运行 Weather MCP Server。 2. Client 成功调用 Server。

✅ Sprint 1 Tasks

- [ ] **Task 1.1 (Env)**: 配置 Python 环境，安装 `mcp[cli]`, `requests`, `python-dotenv`。

- [ ] **Task 1.2 (Server)**: 编写 `weather_server.py` 。使用 FastMCP 定义 `@tool: get_current_weather` 。
- [ ] **Task 1.3 (Client)**: 编写 `agent_client.py` 。实现 Client 连接 Server。
- [ ] **Task 1.4 (Prompt)**: 设计 System Prompt，设定 Agent 的基本人设。

🏃 Sprint 2: The Investor (Market Data)

主题: 引入结构化数据，处理数值准确性。

ID	User Story (用户故事)	Acceptance Criteria (验收标准)
US-03	作为投资者，我希望看到关注股票的涨跌幅。	1. 准确报告收盘价和变化。 2. 数据源实时。
US-04	作为初学者，我希望 AI 给出简单情绪分析。	1. 零幻觉。 2. 使用轻量模型 (Flash) 进行数据总结。

✅ Sprint 2 Tasks

- [ ] **Task 2.1 (Server)**: 扩展 Server，新增工具 `get_stock_price(ticker)` 。
- [ ] **Task 2.2 (Logic)**: 集成 `yfinance` 库，增加异常处理。
- [ ] **Task 2.3 (Routing)**: 优化 Client 逻辑，确保正确调用工具。

🏃 Sprint 3: The Researcher (Config as Resource)

主题: **进阶 MCP 实践** —— 动态偏好管理与搜索。

ID	User Story (用户故事)	Acceptance Criteria (验收标准)
US-05	作为用户，我希望通过对话调整新闻偏好，而不是改代码。	1. 对话："以后别推体育新闻"，Agent 自动更新配置。 2. 下次运行时自动生效。
US-06	作为 MCP 学习者，我希望利用 Resource 机制管理配置。	1. LLM 能读取 `config://preferences` 。 2. LLM 能调用 `update_preferences` 。

✅ Sprint 3 Tasks

- [ ] **Task 3.1 (MCP Resource): (Upgraded)** 在 Server 中实现 Resource `config://preferences`。读取本地 `config.json` 并暴露给 LLM。
- [ ] **Task 3.2 (MCP Tool): (Upgraded)** 实现工具 `update_preferences(key, value, action)`。支持 `add/remove` 操作。
- [ ] **Task 3.3 (Search)**: 实现 `search_web(query)`。**关键逻辑**：在搜索前，代码自动读取 Resource 中的黑名单，过滤 Query 或搜索结果。
- [ ] **Task 3.4 (Scrape)**: 新增网页抓取工具 `fetch_url_content(url)`。

🏃 Sprint 4: The Analyst (Memory & Cost Optimization)

主题: 长期记忆、去重与多模型协作。

ID	User Story (用户故事)	Acceptance Criteria (验收标准)
US-07	作为用户，我不希望看到昨天已经推过的新闻。	1. 系统自动过滤掉过去 3 天已推送的 URL。 2. 对于持续性事件，简报能引用昨日进展。
US-08	作为付费用户，我希望在免费额度内完成任务。	1. 筛选/分类任务强制使用 **Gemini Flash**。 2. 仅最终写作使用 **Gemini Pro**。

✅ Sprint 4 Tasks

- [ ] **Task 4.1 (Memory/State): (New)** 创建 `history.json`。记录 `{date, title_hash, url, summary}`。
- [ ] **Task 4.2 (Context Injection)**: 利用 MCP Prompt 机制 (或 Context Padding)，在生成简报前注入“昨日简报摘要”。
- [ ] **Task 4.3 (The Gatekeeper - Flash)**: 实现分类 Agent (使用 Gemini Flash)：
  - 输入：新闻标题/摘要
  - 输出：`Is_New` (Boolean), `Needs_Verification` (Boolean)
- [ ] **Task 4.4 (The Synthesizer - Pro)**: 只有当 Gatekeeper 认为有价值时，才调用 Pro 模型进行多源对比和最终写作。

🏃 Sprint 5: Automation & Delivery

主题: 全自动化与定时推送。

ID	User Story (用户故事)	Acceptance Criteria (验收标准)
US-09	我希望每天早上 8 点收到整理好的简报。	1. 无需人工干预。

2. 推送成功率 > 99%。

✅ Sprint 5 Tasks

- [ ] **Task 5.1 (Pipeline)**: 编写 `daily_job.py` 。逻辑链：
    1. Load Config (Resource)
    2. Load History (State)
    3. Flash Filter & Verify
    4. Pro Synthesize
    5. Update History
- [ ] **Task 5.2 (Notify)**: 编写推送模块。
- [ ] **Task 5.3 (Deploy)**: 部署 Cron Job。

📝 资源准备 (Resource Checklist)

- [ ] **LLM API Key**: Google AI Studio API Key (确保开通 Flash 和 Pro 的免费 Tier)
- [ ] **OpenWeatherMap API Key**
- [ ] **Search API Key**: (Serper 或 Brave Search 免费层)