AI 学习助手 2.0 - Obsidian 插件实现方案建议

整体方案思路

核心原则：优先考虑集成 Obsidian 社区的现成插件，其次是基于提供类似功能的项目进行调整和二次开发，最后再考虑完全自定义开发。目标是尽量减少开发难度和工作量，快速积累项目经验，快速产出可供社群测试和迭代的早期版本。

1. AI 输入交互优化

目标：提升与 AI 模型交互的便捷性和效率。

1.1 剪贴板优化（合并多个内容，以一定格式提供给 AI）

- **方案**：利用能处理文本和剪贴板内容的 AI 插件，或结合模板插件。
- **推荐插件/思路**：
  - `Mesh AI`: 处理剪贴板内容，集成多 AI 平台，支持自定义处理模式。
  - `Obsidian Knowledge Weaver`: 展开笔记中的嵌入内容并复制到剪贴板，整合信息。
  - 通用文本处理插件或 `Templater` / `Templates` (核心插件): 创建格式模板，手动整理合并后快速格式化。

1.2 以按钮的形式进行多种提示词和上下文的组合

- **方案**：使用内置提示词管理和快捷操作功能的 AI 插件，或宏命令插件。
- **推荐插件/思路**：
  - `BMO Chatbot`: 支持多种 LLM，可配置不同"角色"和命令。
  - `ChatGPT MD`: 若主要使用 OpenAI 模型，提供流畅集成。
  - `Text Generator Plugin`: 定义复杂文本生成模板和指令。
  - `QuickAdd`: 创建宏命令，组合多个操作。

1.3 模型温度快速调整

- **方案**：多数 AI 交互插件允许在设置中调整模型参数。
- **推荐插件/思路**：
  - `BMO Chatbot`, `ChatGPT MD` 等主流 AI 插件。

1.4 上下文消耗显示 & 上下文长度管理

- **方案**：部分高级 AI 插件提供 Token 计数。有效管理依赖智能选择发送内容。
- **推荐插件/思路**：
  - AI 插件如 `BMO Chatbot` 可能显示 Token 信息。
  - `Obsidian Knowledge Weaver`: 手动构建和控制上下文。
  - 超长上下文考虑 RAG 方案 (见 4.3)。

1.5 提示词模版

- **方案**：利用 Obsidian 模板插件或 AI 插件自带模板功能。
- **推荐插件/思路**：

    - `Templater` (社区插件) 或 `Templates` (核心插件)。
    - 多数 AI 插件 (`BMO Chatbot`, `Text Generator Plugin`, `Mesh AI`) 支持保存和复用提示词。

1.6 提示词预判（在用户提问之前，根据情景判断生成）

- **方案**：高级功能，可能需自定义开发。插件分析当前笔记内容、标签等，结合规则推荐。
- **实现思路**：分析当前上下文（笔记内容、标签、最近活动），通过预设规则或简单模型生成建议。

1.7 自动衍生问题，自动挑衅，自动 PUA，回形针小精灵

- **方案**：主要通过精心设计的系统提示词 (System Prompt) 或特定指令实现。
- **推荐插件/思路**：
    - 任何允许自定义系统提示词并与强大 LLM 交互的插件，如 `BMO Chatbot`。效果取决于 Prompt 工程。

2. 语音交互（接入 Obsidian 的操作）

目标：将语音能力深度集成到学习流程中。

2.1 一键本地部署（语音模型）

- **方案**：用户预先在本地配置好语音模型 (如 Whisper.cpp, Ollama)，插件调用本地服务。Obsidian 插件本身难以完全实现复杂环境的"一键部署"。
- **推荐插件/思路**：
    - `obsidian-transcription`: 支持本地 Whisper ASR (需用户自行配置)。
    - 若本地模型提供 API，可通过通用 HTTP 请求插件或自定义代码调用。
    - 辅助脚本或 Docker 镜像可简化用户部署。

2.2 实时收音转录 & 语音笔记

- **方案**：寻找支持录音并转录的插件。"实时"通常指录制完成后立即转录。
- **推荐插件/思路**：
    - `whisper-obsidian-plugin`: 录音/上传音频，使用 OpenAI Whisper API 转录。
    - `obsidian-vox`: 监控文件夹中音频文件并自动转录，计划内置录音。
    - `obsidian-audio-notes`: 配合音频文件和转录稿笔记，计划增强 `Audio recorder` 功能。
    - Obsidian 核心插件 `Audio recorder` (核心插件): 基础录音。

2.3 视频语音转录 / 实时翻译

- **视频转录方案**：
    - `obsidian-transcription`: 支持多种音视频格式 (通过 ffmpeg)，可配合本地 Whisper。
    - `Mesh AI`: 从 YouTube 链接提取转录稿。
    - `obsidian-audio-notes` + `Media Extended`: 处理 YouTube 字幕。
- **实时翻译方案**：通常是"先转录，后翻译"。
    - **推荐翻译插件**: `obsidian-translate` (支持多种在线服务及离线/自托管引擎如 Bergamot, LibreTranslate)。
    - **流程**: STT 插件获取文本 -> `obsidian-translate` 进行翻译。

2.4 语音输出（TTS）

- **方案**：将文本转换为语音朗读。
- **推荐插件/思路**：
  - `obsidian-tts` (by joethei) / `obsidian-text-to-speech` (by AlyceOsbourne): 利用操作系统自带 TTS，支持离线。
  - `obsidian-text2audio`: 使用 Microsoft Azure TTS 服务，可生成音频文件。

2.5 一键生成 AI 总结/润色的有声书

- **方案**：组合功能。1. 内容选定 -> 2. AI 总结/润色 (如 `BMO Chatbot`) -> 3. 文本转音频 (如 `obsidian-text2audio`)。
- **"一键"实现**：
  - `QuickAdd` / `Templater`: 创建宏命令串联插件。
  - 自定义开发：插件内集成 AI API 和 TTS API/库。

3. 学习路线图 -- 游戏化设计

目标：通过游戏化元素提升学习趣味性和动力。

3.1 个人点数面板

- **方案**：插件记录和展示分数/成就。
- **推荐插件/思路**：
  - `obsidian-gamified-tasks` / `Grind Manager` (Grind Manager 是 obsidian-gamified-tasks 的展示名称): 内建金币系统和历史记录。
  - `obsidian-gamified-pkm`: 根据笔记完善度评分、定级、发徽章。
  - `obsidian-achievements`: 添加成就系统，可自定义目标。
  - `Dataview` + `Tracker`: 查询和可视化笔记元数据中的点数。

3.2 应试冲关

- **方案**：将备考内容结构化为任务或挑战。
- **推荐插件/思路**：
  - `obsidian-gamified-tasks` / `Grind Manager`: 将知识点/题目设为带奖励的任务。
  - `Obsidian Tasks (community)`: 管理学习任务。
  - `Spaced Repetition`: 辅助记忆。
  - **思路**：创建"学习关卡"笔记，内含一系列任务。

3.3 学习关卡的衍生交互

- **方案**：最具开放性，可能需较多自定义设计。
- **思路**：
  - **内容解锁**: `Dataview` + 用户等级/点数，动态显示/隐藏下一关卡链接。
  - **AI 互动变化**: 根据用户等级调整喂给 AI 的 Prompt，改变 AI 提问方式或难度。
  - **自定义视图**: 自研插件可使用 Svelte/React 创建丰富交互界面。`Buttons` 插件可创建简单点击交互。

4. 针对线下场景（多人一机）的功能优化

Obsidian 为单用户本地应用，此部分需变通。

4.1 AI 多角色扮演

- **方案**：通过 Prompt 工程实现。
- **推荐插件/思路**：
  - 支持灵活 Prompt 输入的 AI 插件 (`BMO Chatbot`, `ChatGPT MD`, `Mesh AI`)。在 Prompt 中指示 AI 扮演角色。

4.2 多用户输入

- **方案**：Obsidian 不支持原生多账户。
  - **轮流使用，内容区分**: 不同用户在不同笔记/文件夹工作，或同一笔记内用标记区分 (如 `[用户A]: xxx`)。
  - **自定义插件 UI**: 自研插件可设计允许多"角色"轮流在共享 AI 对话框输入的界面。

4.3 超长上下文方案

- **方案**：依赖 AI 模型能力及插件传递上下文的方式。
- **推荐插件/思路**：
  - **选用长上下文模型**: 通过 `BMO Chatbot` 等连接支持长文本的云端模型 (Claude 3, GPT-4-Turbo) 或本地部署大模型 (Ollama + Llama 3 长上下文版)。
  - **RAG (Retrieval Augmented Generation)**:
    - `Smart Connections`: 在库内语义搜索相关笔记片段作上下文，支持本地/云端模型。
    - `Khoj`: 类似个人第二大脑，支持本地文档和网络内容 AI 问答。
    - `Obsidian Knowledge Weaver`: 手动辅助构建上下文。
  - **自定义开发 RAG**: 插件内集成文档切分、向量化、存储和检索。

5. 开发与集成建议

1. **优先组合现有插件**: 充分利用社区插件成熟能力。
2. **利用"胶水"工具**: `QuickAdd` 或 `Templater` 创建宏命令串联不同插件功能。
3. **聚焦自定义开发**: 对现有插件无法满足的独特交互、特定 UI 需求进行自研。
4. **工程实践**:
   - **环境与依赖**: Obsidian 插件开发用 `npm`/`yarn`。Python 后端/脚本可用 `uv` + 清华镜像源。
   - **测试**: 前端 Jest/Vitest；后端据技术栈定。
   - **部署**: 纯前端插件发布至 Obsidian 社区。若涉本地服务 (如本地 AI 模型)，提供清晰部署指南 (Docker 是好选择)。Kubernetes 对此场景可能过重。

行动计划建议

1. **深度试用**: 安装并测试推荐的核心插件，熟悉其功能、配置和局限。
2. **需求映射**: 将 PRD 各功能点与插件能力精确匹配，明确哪些可直接用、组合或自研。
3. **原型验证**: 从核心功能入手，利用现有插件快速搭建可用原型，快速迭代。

此文档旨在根据您提供的 PRD，提供基于 Obsidian 插件生态的实现方案。 核心原则是快速迭代，优先利用现有成熟方案。