Anti-Copilot: 以多代理協作與間隔重複將 IDE 卡關訊號轉化為適應性程式學習路徑

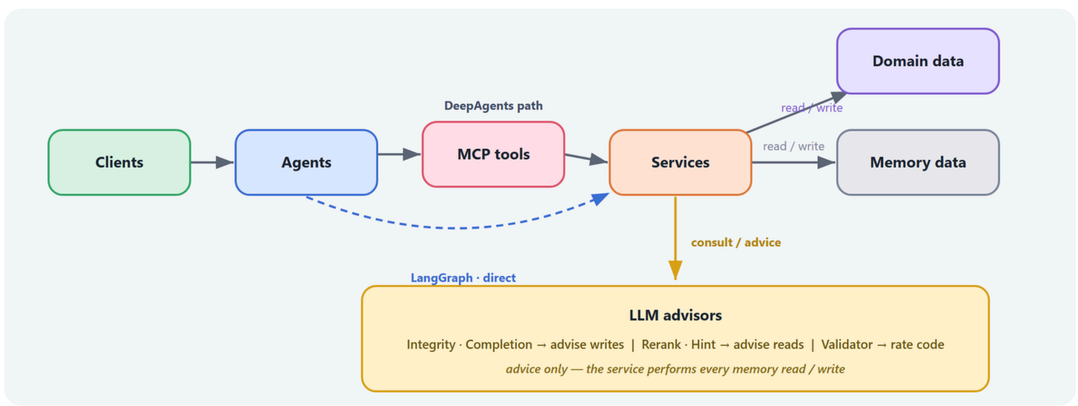
指導教授：莊坤達教授 專題成員：余沛承、陳睿謙、何維展、蘇梓豪

不是代寫程式的 chatbot — 而是把 IDE 卡關訊號 轉換成 個人化學習路徑、Memory 與 FSRS 複習排程 的 **Adaptive Programming Learning System**

研究動機

AI coding tools 能快速產生程式碼，卻容易讓開發者跳過底層理解。本專題提出 Anti-Copilot：一套以「學習」而非「代寫」為核心的 AI 協作式程式學習系統。系統連結使用者的學習目標、實作任務與 IDE 卡關訊號，透過多代理協作、自適應內容生成、structured memory 與 FSRS 間隔重複 機制，將實際卡關轉化為可追蹤、可複習的個人化學習路徑。

系統架構

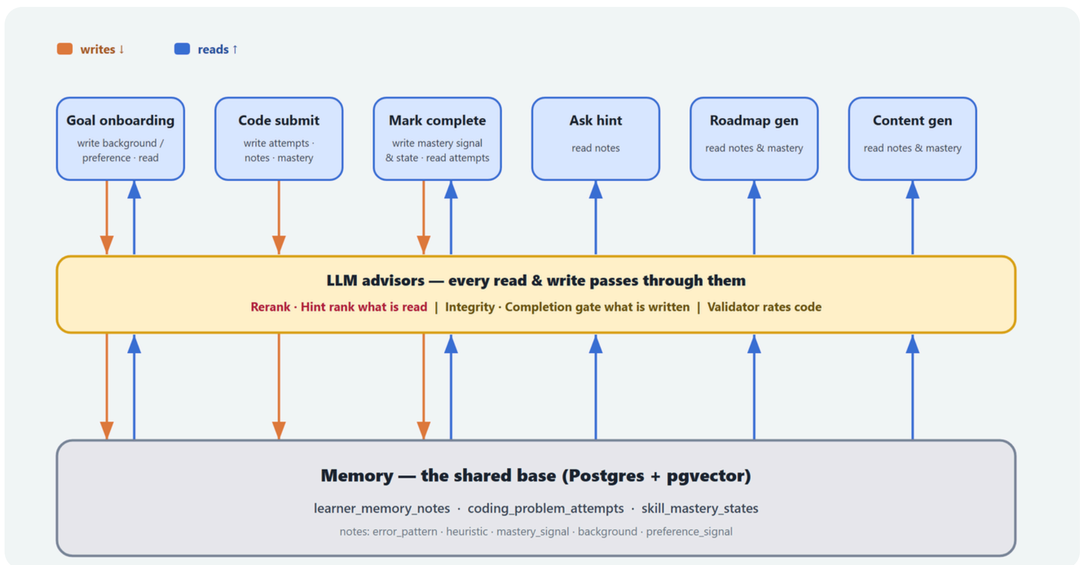


技術與環境

為什麼不用 Markdown-based Memory

主流 AI coding assistant（如 OpenClaw）多以 Markdown 檔案記憶，搭配 Embedding + TTL 強化，在單人、短期任務上直觀且有效。但在多用戶、長期運作的產品中會暴露出檔案系統的根本限制：

Markdown 本質上是 append-only。檔案一多，agent 難以定位要修改的那一條資訊；不同檔案對同一件事的矛盾描述也無法自動解決，導致 reasoning 污染、token 浪費、記憶遺失

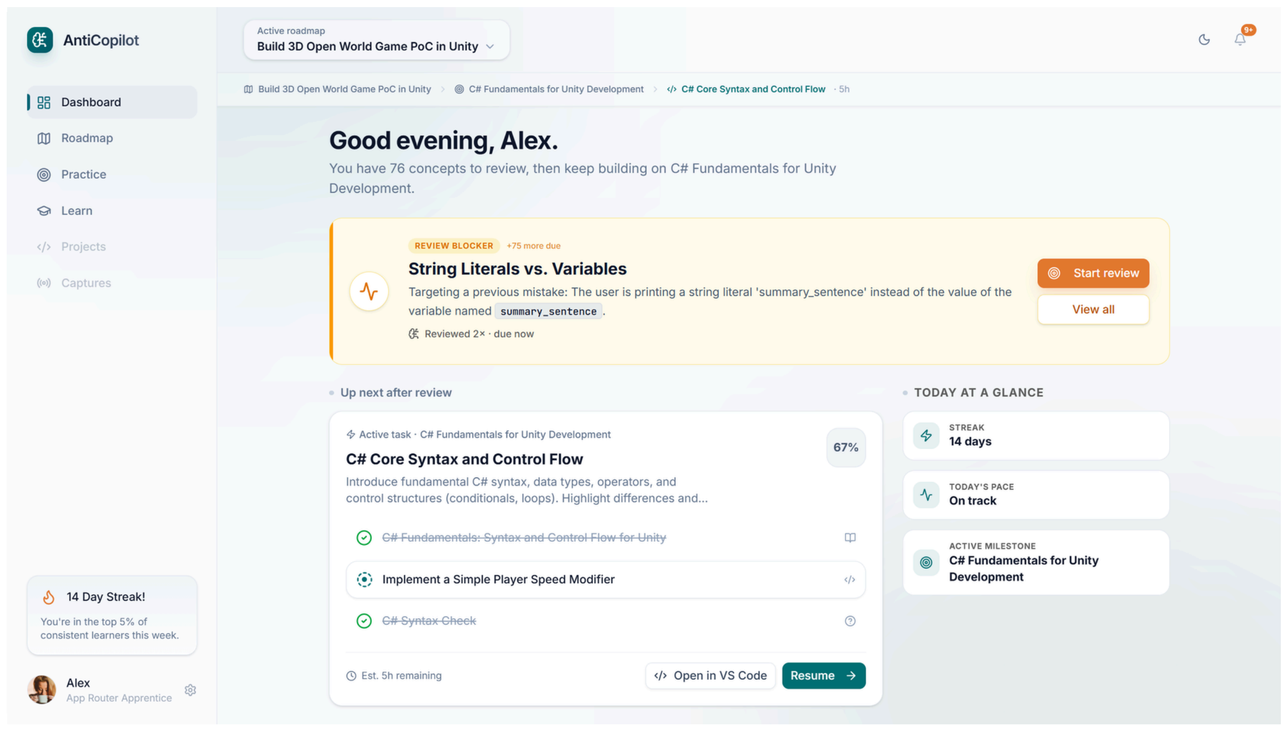


我們的做法

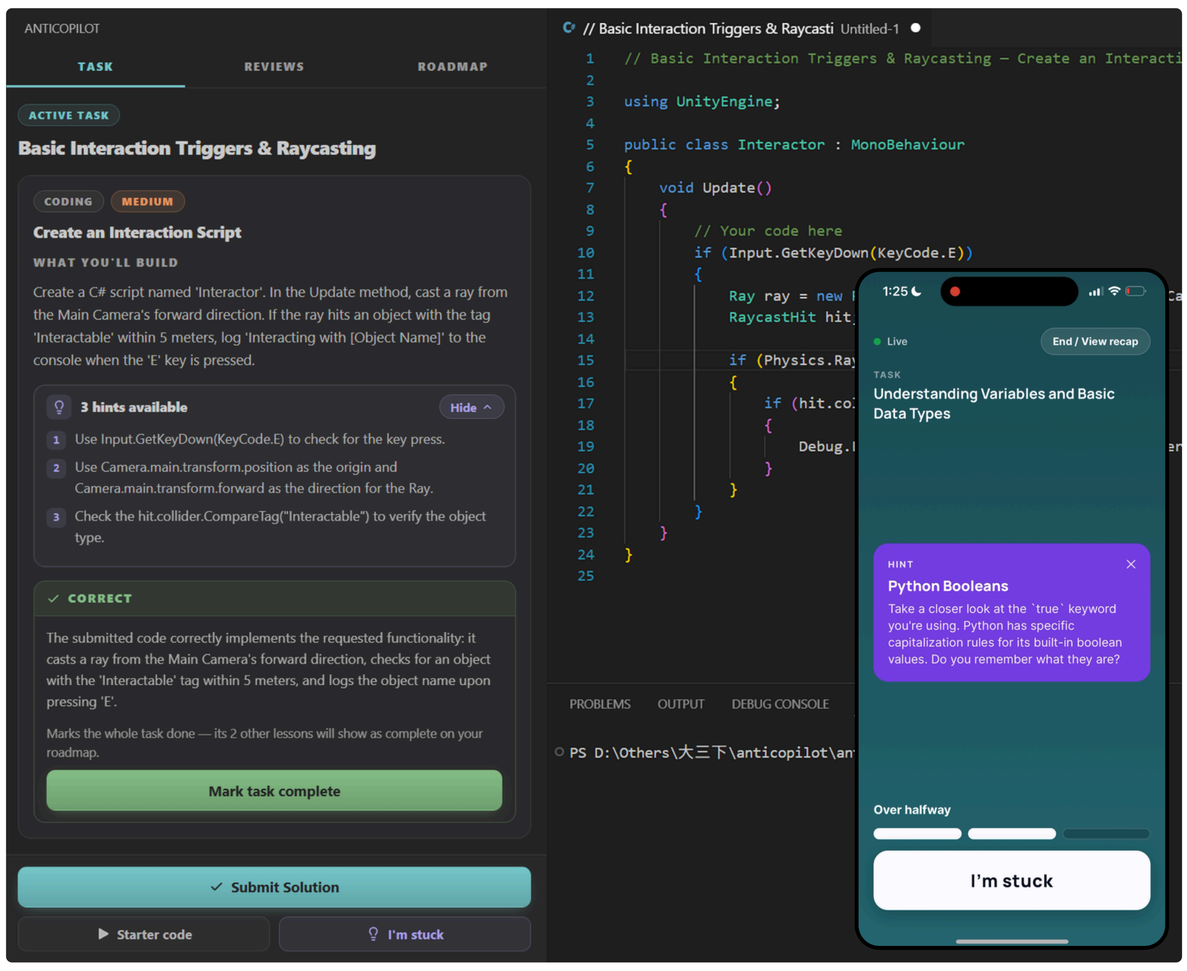
面對多用戶、持續累積的記憶量，需要 database-level 的管理機制：

- 寫入時主動做狀態判斷，防止矛盾共存 — 參考 **Mem0** (2504.19413) 的 write-time conflict resolution
- 記憶單元結構化並 linked 到 skillpath，非純文字 — 參考 **A-MEM** (2502.12110) 的 atomic + linked note 設計
- code attempt evidence 與 long-lived notes 分離，防止短期雜訊污染長期記憶 — 參考 **GAM** (2604.12285) 的 encoding/consolidation 解耦
- 讀取時以 purpose-specific LLM Rerank Advisor 精選 candidates，拒絕高召回低精確的雜訊— 參考 **RMM** (2503.08026) 的 Retrospective Reflection

成果展示

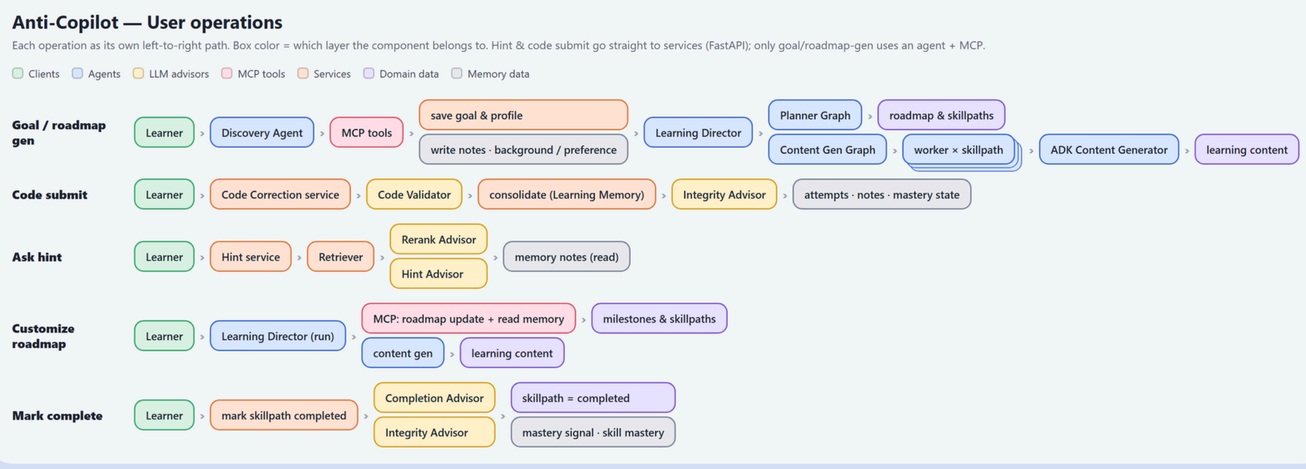


Frontend Interface 以 review blocker 與進度總覽引導使用者下一步學習；同時涵蓋 AI 對話式 roadmap 生成與客製化學習內容、 FSRS 間隔重複的複習佇列



VSCode Extension + App VS Code 側邊欄顯示任務說明,並在編輯器開啟 starter code; Live App 為即時反饋介面,顯示任務完成度、提示使用者卡關狀態,並提供 AI Hint

Demo Flow



核心亮點

Structured learner memory
+
FSRS spaced repetition

讓每一次卡關都成為下一次複習的起點

在真實 coding 情境中逐步補足知識缺口

專題連結