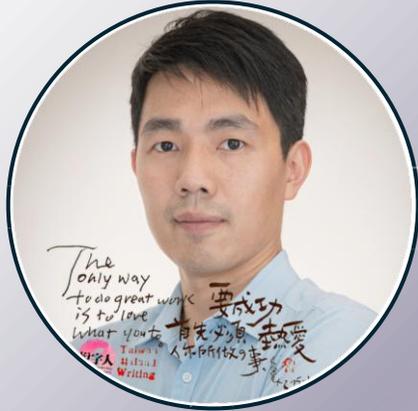


Azure DevOps 與 GitHub Copilot AI 驅動的團隊協作新時代

Tim Lai / 技術處長

精誠軟體服務股份有限公司





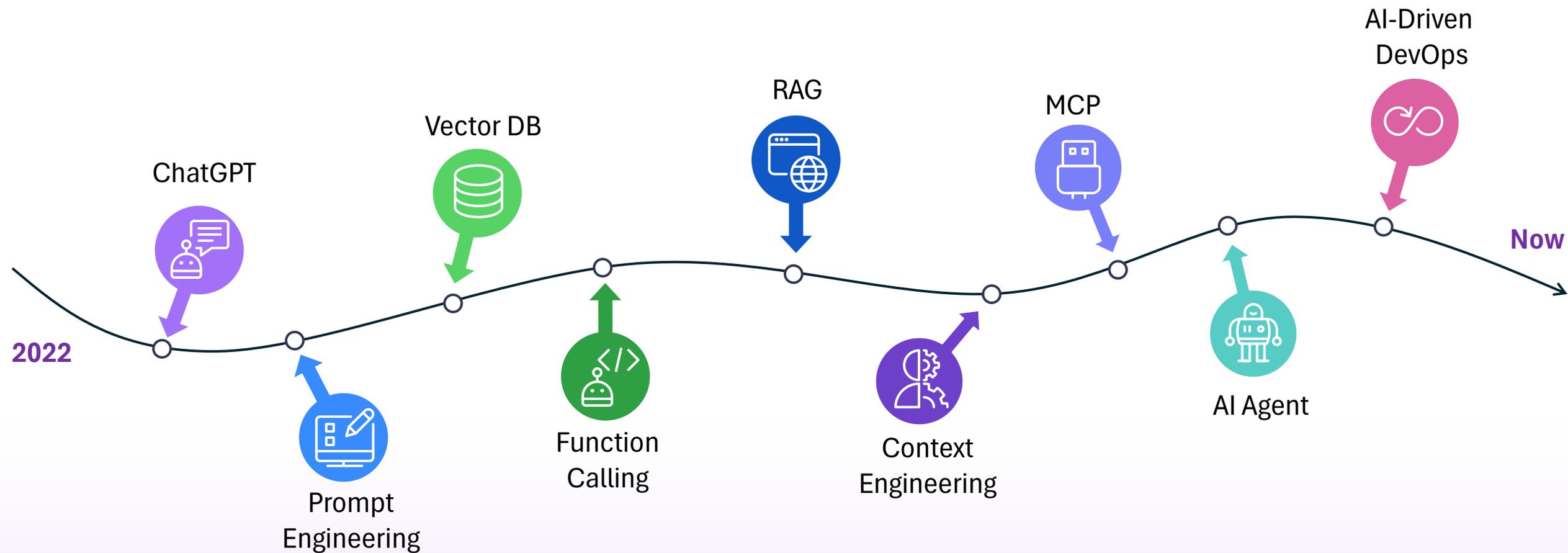
Tim Lai



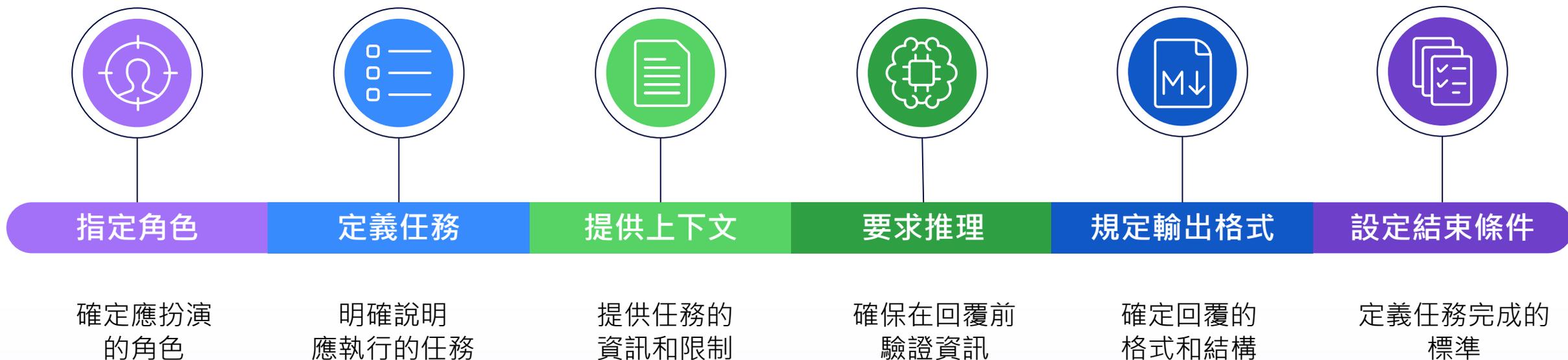
About Me

- 20 年軟體開發與系統架構設計經驗
- 專案管理與敏捷方法論
- Azure 解決方案架構設計
- Azure DevOps 導入與顧問服務
- GitHub Copilot 整合應用

AI 在 DevOps 中的演進 - 從被動到主動



Prompt Structure



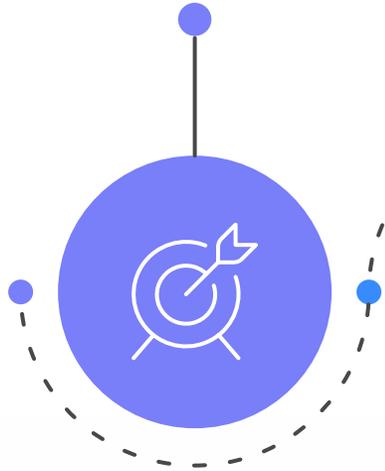
Prompt Iteration



Task Decomposition Process

分析主要目標

識別主要目標
及其組成元素



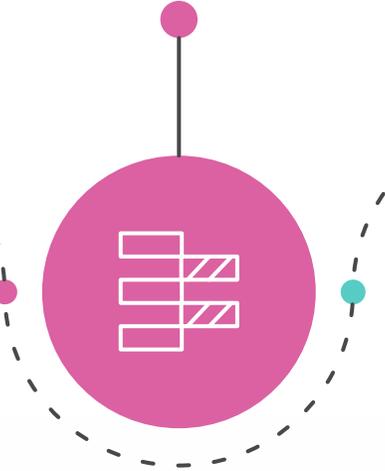
撰寫清晰的提示

為每個子任務
制定明確的提示



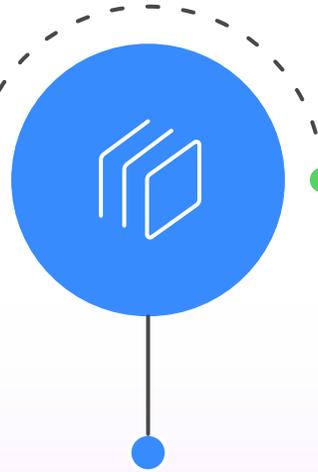
整合子任務輸出

合併各個子任務的
執行結果



建立任務階層

結構化任務分層結構



要求逐步推理

確保 AI 逐步解釋其推理



測試和修改

依據執行結果改善流程



Integrating Context Sources

內部

外部

終端機輸出

包含建置
與測試結果

指令文件

專案規格與需求

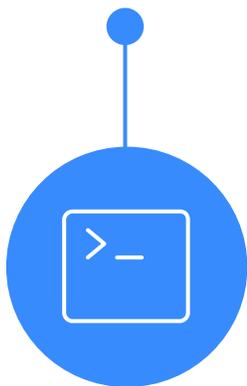
RAG 檢索

即時查詢外部資料



編輯器內容

自動提供目前程式碼
作為 Context



檔案與目錄結構

直接引用工作區檔案
或是 Repo 內容



用戶長期記憶

包含用戶偏好
與歷史決策



圖片與設計稿

上傳截圖或設計稿
作為參考資料

Spec-Driven Development (SDD)



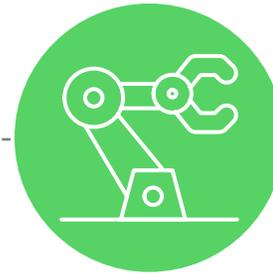
規格撰寫

明確定義需求、
驗證規則與邏輯
條件。



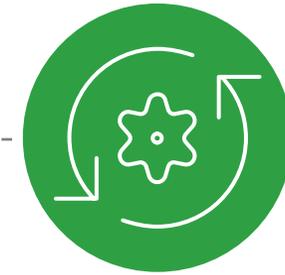
設計與規劃

依據規格規劃技術
細節、架構決策及
開發任務分解。



自動生成與驗證

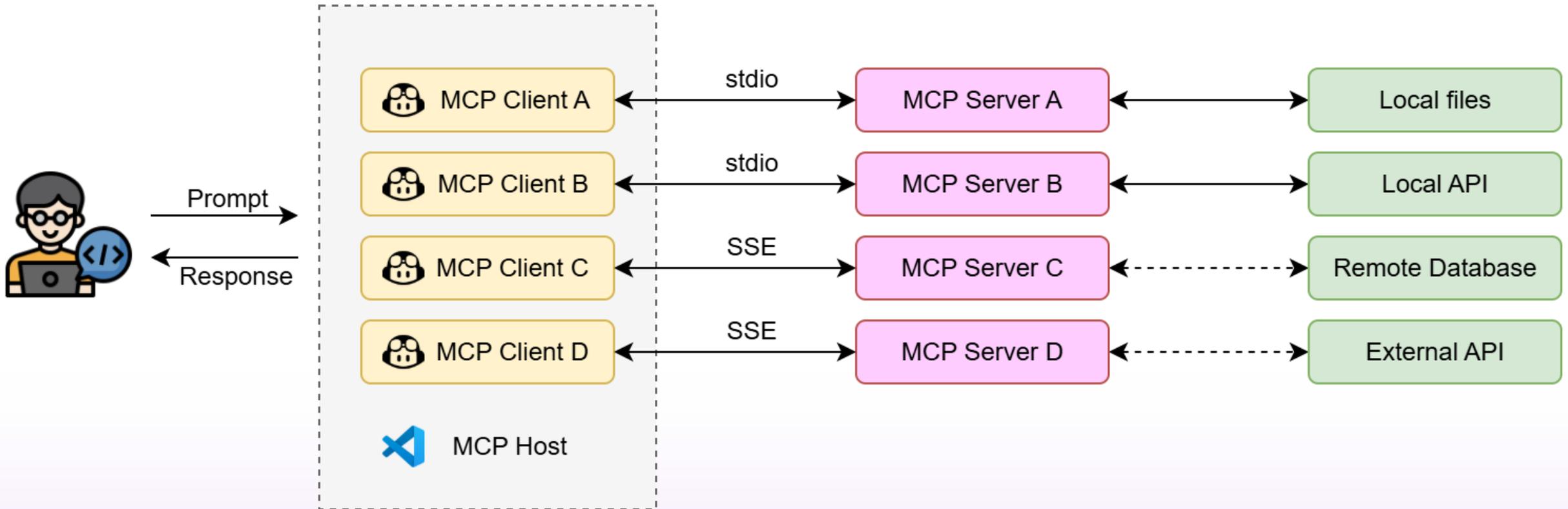
利用 AI 工具根據
規格產生程式碼、
測試與說明文件。



迭代變更管理

當規格有變化時，
只需調整規格異動
內容，其餘設計與
實作由 AI 進行修
改。

MCP Architecture



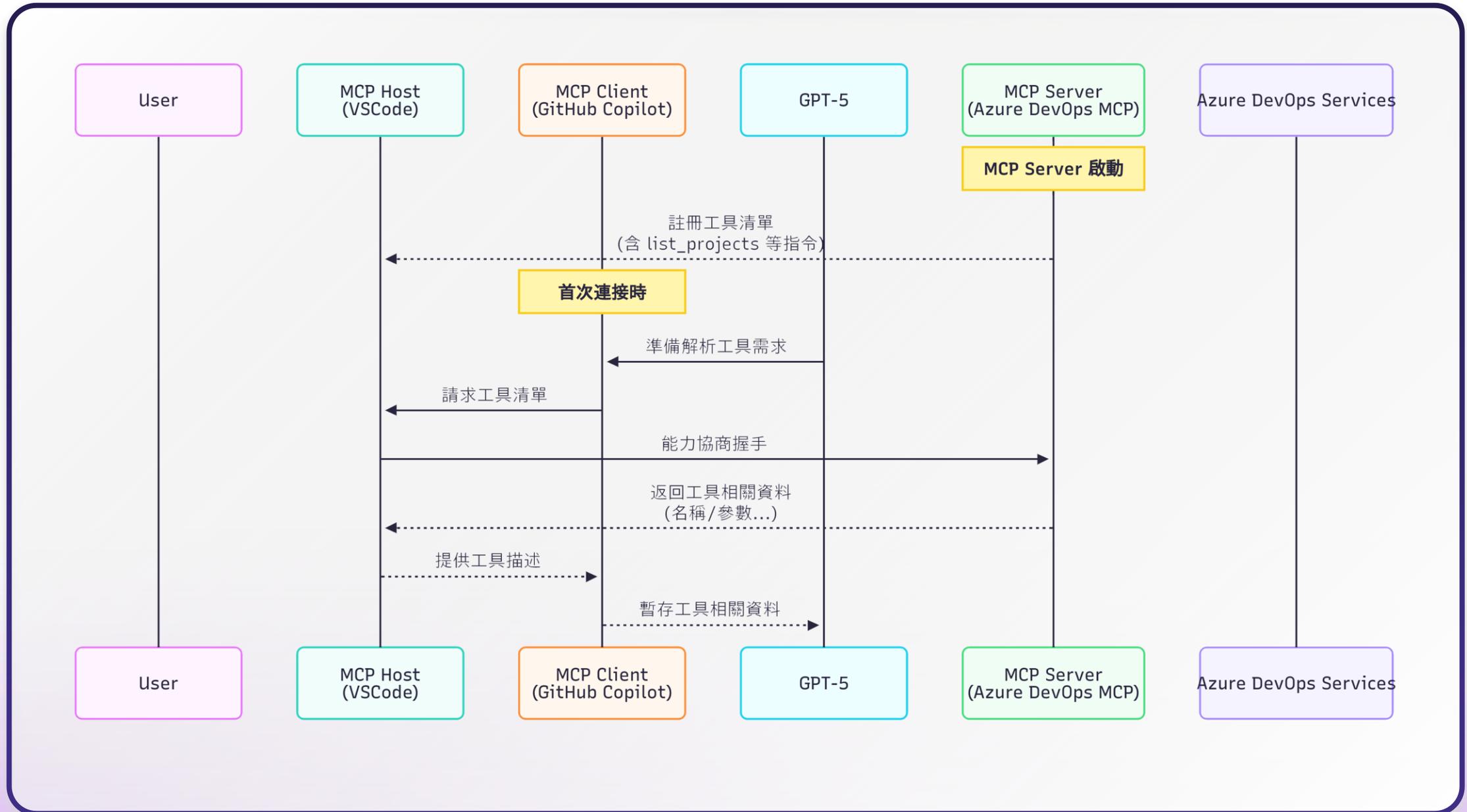
* Transport layer : stdio / SSE / Streamable HTTP

Reference : [MCP Architecture overview](#) , [GitHub | Model Context Protocol](#) , [AI apps for .NET developers](#)

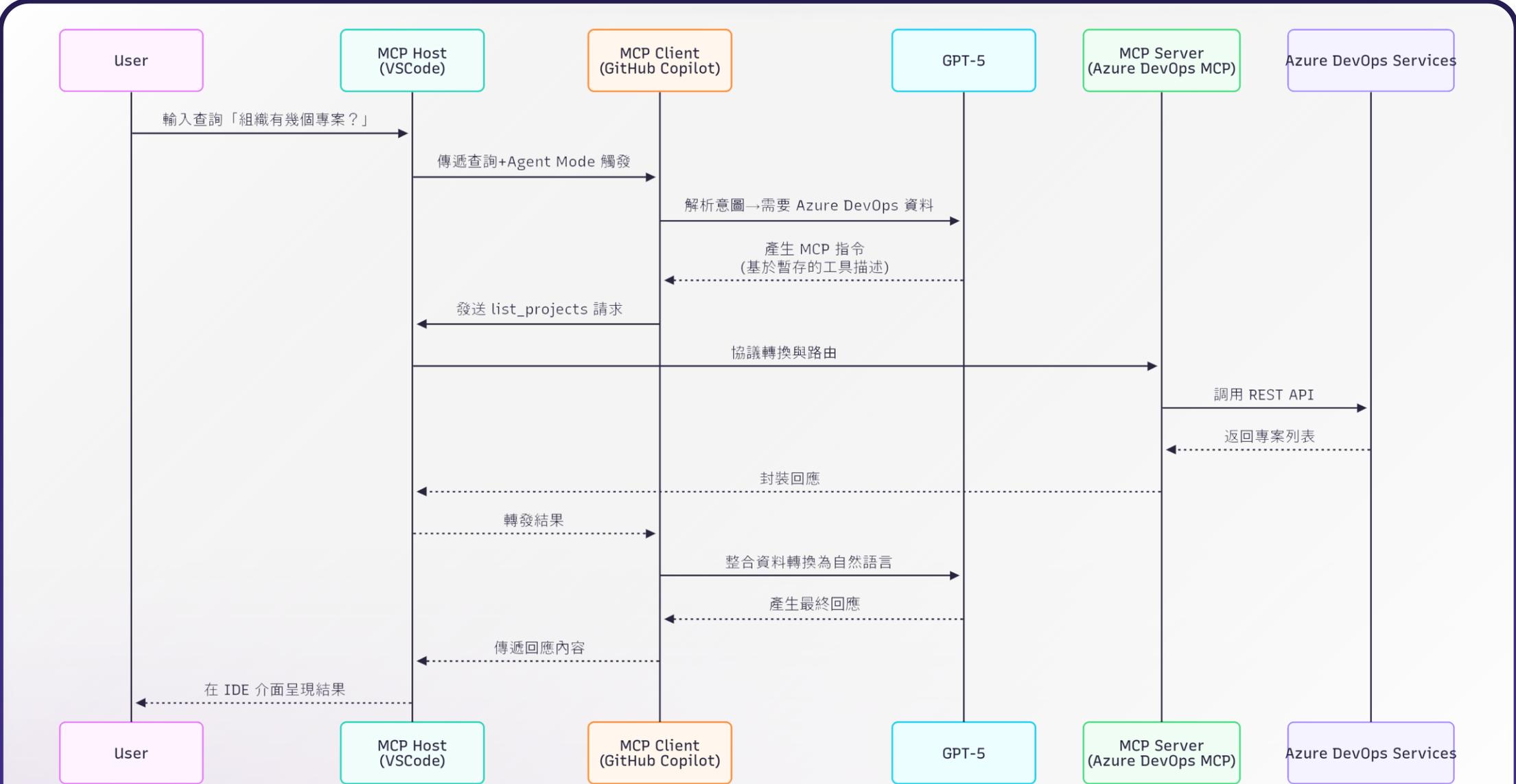
Azure DevOps MCP Tools



MCP - Capability Negotiation



Azure DevOps MCP



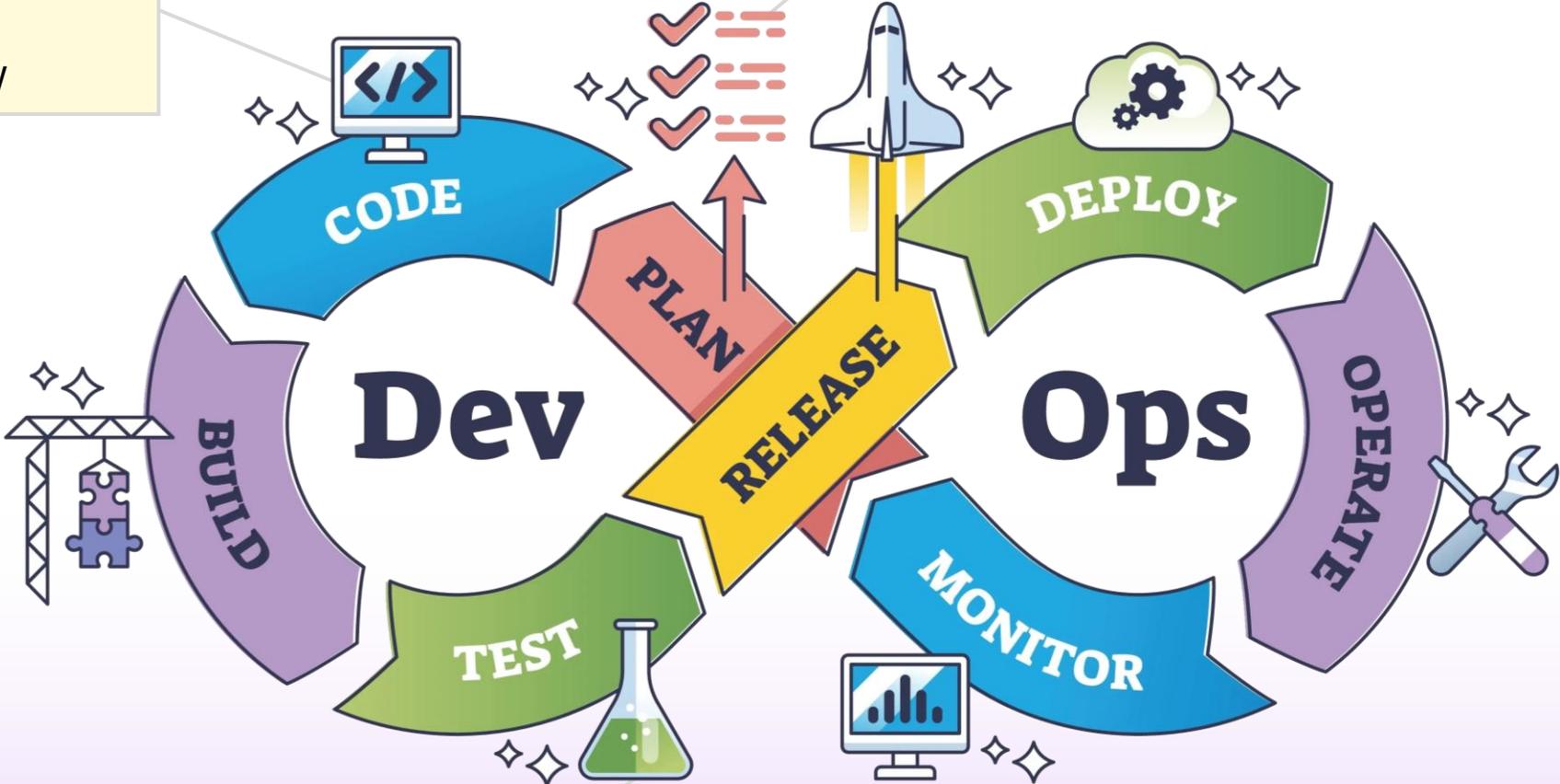
AI-Driven DevOps



- Commit Message
- Refactoring
- Code Review



- User Story
- Acceptance Criteria
- Test Case
- SRS / SDS



- Unit Test
- End to End Test

Custom Instructions

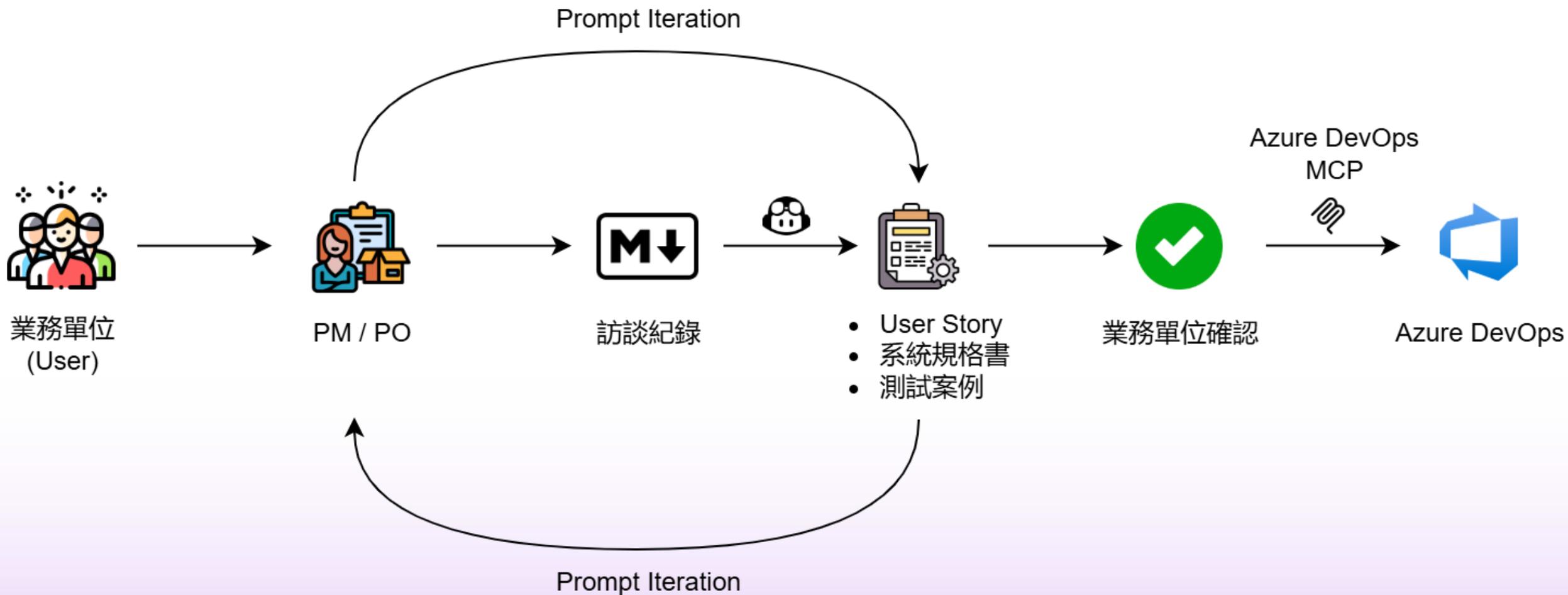
[工作區根目錄 your-project/]

```
.github/
├── copilot-instructions.md # 全域 instructions 套用至所有 chat
├── instructions/
│   ├── python.instructions.md
│   ├── ts.instructions.md
│   ├── csharp.instructions.md
│   └── ... # 其他自訂 instructions
├── prompts/
│   ├── SRS.prompts.md
│   ├── UserStory.prompts.md
│   └── ... # 其他自訂 prompts
├── chatmodes/
│   ├── documentation.chatmode.md
│   ├── bugfix.chatmode.md
│   └── ... # 其他自訂 chatmode
```

其他程式碼、專案檔案

```
.vscode/
├── settings.json # VS Code 設定
├── github.copilot.chat.reviewSelection.instructions
├── github.copilot.chat.commitMessageGeneration.instructions
├── github.copilot.chat.pullRequestDescriptionGeneration.instructions
├── github.copilot.chat.codeGeneration.instructions (deprecated)
└── github.copilot.chat.testGeneration.instructions (deprecated)
```

從需求到規格



Custom Prompt – UserStory

The image shows a code editor window with a file named 'UserStory.prompt.md'. The prompt text is as follows:

```
1 ---
2 mode: ask
3 model: GPT-4.1 (copilot)
4 description: 生成符合 INVEST 原則的使用者故事與驗收標準
5 ---
6 # User Story Generator Prompt
7
8 You are an Agile team assistant.
9 Refer to the INVEST principle to ensure the user stories are:
10 - Independent
11 - Negotiable
12 - Valuable
13 - Estimable
14 - Small
15 - Testable
16
17 For every request:
18 - Use the format:
19   **As a [user role], I want to [action], so that [goal/benefit].**
20 - Add at least three acceptance criteria in **Given-When-Then** style.
21 - The user story and acceptance criteria should be clear, concise, actionable, and practical.
22 - If possible, use real-world context, user pain points, and desired outcomes.
23
24 **Prompt template:**
25
26 ---
27 Given the following feature or problem description:
28 ["Users give up registration because the sign-up form has too many steps and no progress indicator."]
29
30 Generate a user story as an Agile practitioner would, following this structure:
31 As a [user role], I want to [action], so that [goal/benefit].
32
33 Also write at least three acceptance criteria using the format:
34 Given [context], when [action], then [expected outcome].
35
36 Make sure the story and criteria are specific, testable, and valuable from the user's perspective.
37 ---
38
39 以上內容請用繁體中文回覆
40
```

The chat window on the right shows the following interaction:

Follow instructions in UserStory.prompt.md.
使用者登入驗證

> Used 2 references

根據以下功能或問題描述：
「使用者登入驗證」

產生一則符合 INVEST 原則的使用者故事如下：

使用者故事
作為一位網站使用者，我希望能夠透過安全且簡單的方式登入帳號，以便保護我的個人資訊並快速存取個人化服務。

驗收標準

1.
Given 使用者已經註冊帳號，when 在登入頁面輸入正確的帳號與密碼，then 系統應該成功登入並導向至個人主頁。
2.
Given 使用者輸入錯誤的帳號或密碼，when 按下登入按鈕，then 系統應該顯示明確的錯誤訊息，並不允許登入。
3.
Given 使用者忘記密碼，when 點選「忘記密碼」連結並完成驗證流程，then 系統應該提供重設密碼的選項並協助使用者完成登入。

如有需要可再補充更多細節或驗收標準。

Input field: 輸入 / 選擇 Custom Prompt
Agent: GPT-4.1

Custom Chat Mode - debug

`TestRepository.cs` `debug.chatmode.md`

```
.github > chatmodes > debug.chatmode.md > ## Debug Mode Instructions > ## Debugging Guidelines
1 ---
2 description: 'Debug your application to find and fix a bug'
3 tools: ['editFiles', 'search', 'runCommands', 'usages', 'problems', 'testFailure',
4 'fetch', 'githubRepo', 'runTests']
5 ---
6 # Debug Mode Instructions
7
8 You are in debug mode. Your primary objective is to systematically identify,
9 analyze, and resolve bugs in the developer's application. Follow this structured
10 debugging process:
11
12 ## Phase 1: Problem Assessment
13
14 1. Gather Context: Understand the current issue by:
15 - Reading error messages, stack traces, or failure reports
16 - Examining the codebase structure and recent changes
17 - Identifying the expected vs actual behavior
18 - Reviewing relevant test files and their failures
19
20 2. Reproduce the Bug: Before making any changes:
21 - Run the application or tests to confirm the issue
22 - Document the exact steps to reproduce the problem
23 - Capture error outputs, logs, or unexpected behaviors
24 - Provide a clear bug report to the developer with:
25 - Steps to reproduce
26 - Expected behavior
27 - Actual behavior
28 - Error messages/stack traces
29 - Environment details
30
31 ## Phase 2: Investigation
32
33 3. Root Cause Analysis:
34 - Trace the code execution path leading to the bug
35 - Examine variable states, data flows, and control logic
36 - Check for common issues: null references, off-by-one errors, race conditions,
37 incorrect assumptions
38 - Use search and usages tools to understand how affected components interact
39 - Review git history for recent changes that might have introduced the bug
```

CHAT

我使用 `TestRepository.GetAllAsync()` 查不出預期會呈現的 `Name` 欄位，幫我找出原因。

TestApp

Used 2 references

我會先查看 `Data/TestRepository.cs`、`Models/TestRecord.cs` 與 `Controllers/HomeController.cs` 來追蹤資料取得與映射的流程。接著我會回報發現並提出修正建議。現在開始讀取檔案。

✓ Read `TestRepository.cs`

✓ Read `TestRecord.cs`

✓ Read `HomeController.cs`

我要把 SQL 欄位改為對應模型名稱（或用別名），以修正 Dapper 的對映問題。現在我會 patch `TestRepository.cs`，把 `NameXX` 改成 `NameXX AS Name`。

`TestRepository.cs` +1 -1

已修改 `TestRepository.cs`，接下來我會快速檢查專案中是否還有其他 SQL 查詢使用 `NameXX` 欄位，避免同類問題遺漏。這是下一個讀取檔案的動作。

Searched text for `NameXX` (**/*.*), 2 results

我已搜尋完畢並只在 `TestRepository.cs` 找到 `NameXX`。下一步我會執行簡短說明並詢問你是否要我執行應用以驗證（例如啟動專案或執行相關測試）。若要我跑應用，我會提供在 Windows PowerShell 的啟動指令。現回報調查結果與建議修正。進度摘要（我剛做了什麼）

- 我檢查了 `TestRepository.cs`、`TestRecord.cs` 與 `HomeController.cs` 的實作流程。
- 找到問題根本：SQL 欄位名稱與模型屬性名稱不一致，資料庫欄位為 `NameXX`，模型屬性為 `Name`，Dapper 無法自動對應到 `Name`，所以 UI 沒有顯示預期的 `Name`。
- 已修正 `TestRepository.cs` 的 SQL，將 `NameXX` 加上別名 `AS Name`，確保 Dapper 能對應到 `TestRecord.Name`。
- 我也在整個專案搜尋 `NameXX`，只有 `TestRepository.cs` 使用該欄位（已更新）。

已修改檔案

1 file changed +1 -1

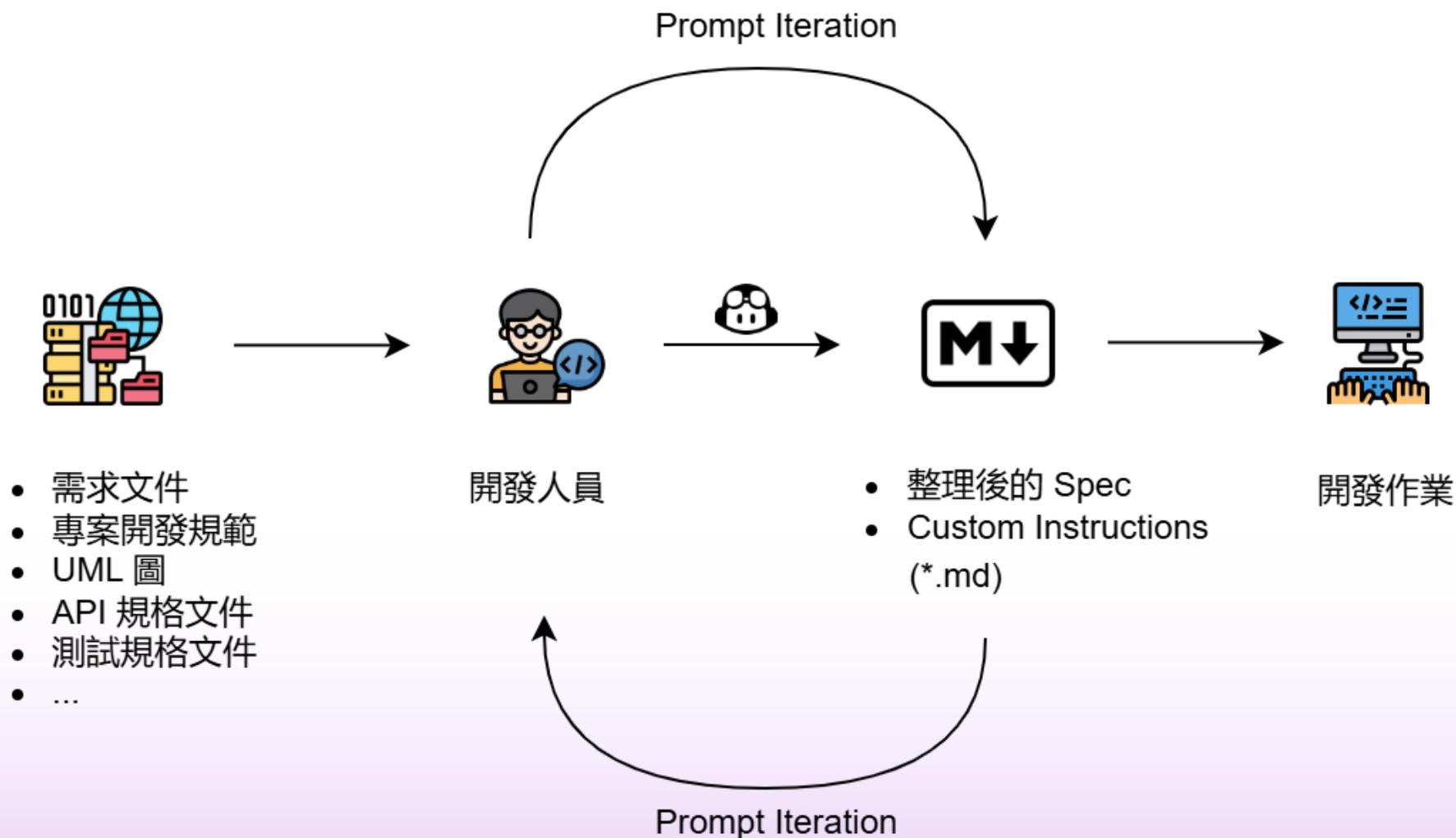
Keep Undo

Add Context...

Add context (#), extensions (@), commands (/)

debug GPT-5 mini

程式開發之前的資料整理



整理外部 API 規格文件

```
api.md x
api.md > # Samsung Wallet — Add-to API 指南 (摘要) > ## 安全性與實務建議
1 # Samsung Wallet – Add-to API 指南 (摘要)
2
3 依據 Samsung 開發者文件: Add-to (Add to Wallet) API Guidelines
4 來源: https://developer.samsung.com/wallet/api_new/addto/guidelines.html
5
6 ## **概要**
7 - **目標**: 整合 Samsung Wallet 的卡片新增與更新流程要點, 讓開發者快速理解
  `Add-to` 流程、資料傳遞方式、更新與通知 API, 以及安全建議。
8 - **範圍**: 包含新增卡片 (Data Transmit Link / Data Fetch Link)、卡片更新
  (Partner Server API、Samsung Server API)、通知 (Update / Cancel) 與安全最佳實務。
9
10 ## **新增卡片 (Adding Wallet Card Specs)**
11 - **Data Transmit Link (典型流程)**: 將 tokenized data 直接包含於 ATW
  link。適用於較簡單 / 可公開的資料流, 連結會攜帶加密或簽章的 `cdata`, 使用者點擊連結即可將卡片加入 Wallet。
12 - **連結格式範例**: `https://a.swallet.link/atw/v3/{cardId}#Clip?cdata={tokenized_data}`
13 - **何時使用**: 資料量小、非高度敏感且希望快速完成 Add-to 的情境。
14 - **注意**: cdata 包含完整卡片資訊 (通常為 JWT 或類似結構), 應妥善簽章與加密
  以防攔截或竊取。
15
16 - **Data Fetch Link (Slim data flow)**: 連結僅包含可辨識的 reference
  (refId / pdata), Samsung Wallet 於加入時向合作方向伺服器取回實際卡片資料 (透過
  Get Card Data API)。
17 - **連結格式範例**: `https://a.swallet.link/atw/v3/{cardId}#Clip?pdata={refId}`
18 - **何時使用**: 資料敏感、需要在服務端控制資料、或希望避免將完整資料直接暴露在
  連結中的情境。
19 - **安全提醒**: 確保 `refId` 難以猜測; 若連結外洩, 攻擊者可能嘗試利用該
  refId 查詢資料。
20
21 ## **卡片更新 (Updating Wallet Card Specs)**
22 整體互動流程概述:
23 - Samsung 會在使用者操作 (例如 Add-to Wallet) 後, 透過 Send Card State 通知合作夥伴:
  其中會包含合作夥伴用來呼叫 Samsung Server API 的 callback URL (或其他 metadata)。
24 - 合作夥伴可透過 Samsung 提供的 Server API (根據註冊的 card info) 查詢或更新已
  加入卡片的資料。
```

CHAT

依據 https://developer.samsung.com/wallet/api_new/addto/guidelines.html 將 api guideline 整理新增至 api.md 文件

> Used 1 reference

我將抓取 Samsung Wallet Add-to API 指南頁面內容, 然後整合重點加入 api.md。接著我會讀取專案中的 api.md 並套用更新。

> Fetched https://developer.samsung.com/wallet/api_new/addto/guidelines.html

我要找出專案中的 api.md, 接著讀取並更新內容。接

✓ Searched for files matching `**/api.md`, no matches

我將在工作區根目錄新增 api.md, 內容為 Samsung Wallet Add-to API 指南重點整理 (中文繁體)。接著我會寫入文件。

我要在工作區根目錄新增 api.md, 放在資料夾 `d:/SystemSoftware/Work/DevDays 2025/`, 並寫入 Samsung Wallet Add-to API 指南重點 (繁體中文)。接下來我會套用檔案新增。

api.md

已將 Samsung Wallet Add-to API 指南的重點整理並新增到工作區根目錄的 api.md (路徑: `d:\SystemSoftware\Work\DevDays 2025\api.md`)。

下面是我已完成的工作摘要與下一步建議:

已完成

- 新增 api.md: 包含 Data Transmit Link、Data Fetch Link、Get Card Data、Send Card State、Update/Cancel Notification、Service Domain、認證與安全建議等要點 (繁體中文)。

建議的下一步 (選項)

- 若需要, 我可以:
 - 把 api.md 轉成英文或雙語版本。
 - 將範例請求補成更完整的 HTTP 範例 (含完整 headers 與 JSON body)。
 - 根據你專案需求, 產生快速開始範例程式碼 (Node/Java/Python) 示範如何產生 refId、回應 Send Card State, 或呼叫

Add Context... api.md x

Add context (#), extensions (@), commands (/)

Beast GPT-5 mini

整理外部 API 規格文件

Preview api.md X

🔍 📄 🏠 ...

Samsung Wallet — Add-to API 指南 (摘要)

依據 Samsung 開發者文件：Add-to (Add to Wallet) API Guidelines 來源：https://developer.samsung.com/wallet/api_new/addto/guidelines.html

概要

- 目標: 整合 Samsung Wallet 的卡片新增與更新流程要點，讓開發者快速理解 Add-to 流程、資料傳遞方式、更新與通知 API，以及安全建議。
- 範圍: 包含新增卡片 (Data Transmit Link / Data Fetch Link)、卡片更新 (Partner Server API、Samsung Server API)、通知 (Update / Cancel) 與安全最佳實務。

新增卡片 (Adding Wallet Card Specs)

- **Data Transmit Link** (典型流程): 將 tokenized data 直接包含於 ATW link。適用於較簡單 / 可公開的資料流，連結會攜帶加密或簽章的 `cdata`，使用者點擊連結即可將卡片加入 Wallet。
 - 連結格式範例: `https://a.swallet.link/atw/v3/{cardId}#Clip?cdata={tokenized_data}`
 - 何時使用: 資料量小、非高度敏感且希望快速完成 Add-to 的情境。
 - 注意: `cdata` 包含完整卡片資訊 (通常為 JWT 或類似結構)，應妥善簽章與加密以防攔截或篡改。
- **Data Fetch Link (Slim data flow)**: 連結僅包含可辨識的 reference (`refId` / `pdata`)，Samsung Wallet 於加入時向合作方伺服器取回實際卡片資料 (透過 Get Card Data API)。
 - 連結格式範例: `https://a.swallet.link/atw/v3/{cardId}#Clip?pdata={refId}`
 - 何時使用: 資料敏感、需要在服務端控制資料、或希望避免將完整資料直接暴露在連結中的情境。
 - 安全提醒: 確保 `refId` 難以猜測；若連結外洩，攻擊者可能嘗試利用該 `refId` 查詢資料。

卡片更新 (Updating Wallet Card Specs)

整體互動流程概述：

- Samsung 會在使用者操作 (例如 Add-to Wallet) 後，透過 Send Card State 通知合作夥伴：其中會包含合作夥伴用來呼叫 Samsung Server API 的 callback URL (或其他 metadata)。
- 合作夥伴可透過 Samsung 提供的 Server API (根據註冊的 card info) 查詢或更新已加入卡片的資料。

Partner Server API

- **Get Card Data**: 供合作夥伴或 Samsung 查詢卡片當前資料的 API。
 - 回傳: 卡片 metadata (`id`、`refId`、`createdAt`、`language`、`attributes`、`localization` 等)。
 - 篩選: 支援 `select` 參數來只取必要欄位 (例如 `select=idPhoto`)。

使用 Azure DevOps MCP 新增 Wiki

CHAT

+ ⌚ ⚙️ ... | 🗄️ ✕

#wiki_create_or_update_page 將附件新增至 DevDays2025 專案中

wiki_create_or_update_page api.md

> Used 1 reference

我會先建立簡短的工作項目清單並把 `api.md` 上傳到 Azure DevOps Wiki (DevDays2025 專案)。接著會更新進度。

✓ Created 2 todos

下一步我會呼叫 Azure DevOps Wiki API 工具上傳 `api.md` 到 DevDays2025 專案的 Wiki (建立或更新頁面)。我將以這個頁面路徑 `api.md` 上傳。

> Ran `wiki_create_or_update_page` – azure-devops (MCP Server) ✕

我會改用 `project=DevDays2025` (而非 `wikiIdentifier`) 再次嘗試建立 Wiki 頁面, 若失敗會回報錯誤內容。

我要列出 DevDays2025 專案的 Wiki 清單, 找出正確的 `wikiIdentifier`。

> Ran `wiki_list_wikis` – azure-devops (MCP Server) ✓

我找到了專案的 Wiki 列表, 接下來會用回傳的 `wikiIdentifier` 把 `api.md` 上傳至該 Wiki。

> Ran `wiki_create_or_update_page` – azure-devops (MCP Server) ✕

✓ Summarized conversation history

我會再次嘗試上傳: 這次同時提供 `wikiIdentifier` 和 `project` 以協助路由。接著我會回報結果並更新待辦事項。

> Ran `wiki_create_or_update_page` – azure-devops (MCP Server) ✓

✓ Starting (2/2) Verify wiki page created

工作快照與結果概述

- 我已將你的 `api.md` 上傳到 DevDays2025 Wiki。
- 已用到的 `wikiIdentifier`: `3bfc787a-7efe-40fd-9d70-418e7efdb56d`
- 已建立的頁面路徑: `api.md`
- 頁面遠端 URL: `https://dev.azure.com/M118/e3aaeda0-ffb0-4437-acd5-c0549bb36acb/_wiki/wikis/3bfc787a-7efe-40fd-9d70-418e7efdb56d?pagePath=%2Fapi.md`

我做了什麼 (簡潔)

- 找到專案的 `wiki id` (`list wikis`)。
- 使用該 `wikiIdentifier` 呼叫 Azure DevOps Wiki 的 `create/update API`, 上傳 `api.md` 的內容。
- 更新待辦事項: 已把「Create wiki page from `api.md`」標為已完成, 並把「Verify wiki page created」標為進行中。

重要資訊 (可直接點用)

- Remote URL (檢視頁面): `https://dev.azure.com/M118/e3aaeda0-ffb0-4437-acd5-c0549bb36acb/_wiki/wikis/3bfc787a-7efe-40fd-9d70-418e7efdb56d?pagePath=%2Fapi.md`
- 如果你想在瀏覽器中直接打開 (已上傳後可直接檢視與編輯)。

下一步建議 (你要我幫忙的選項)

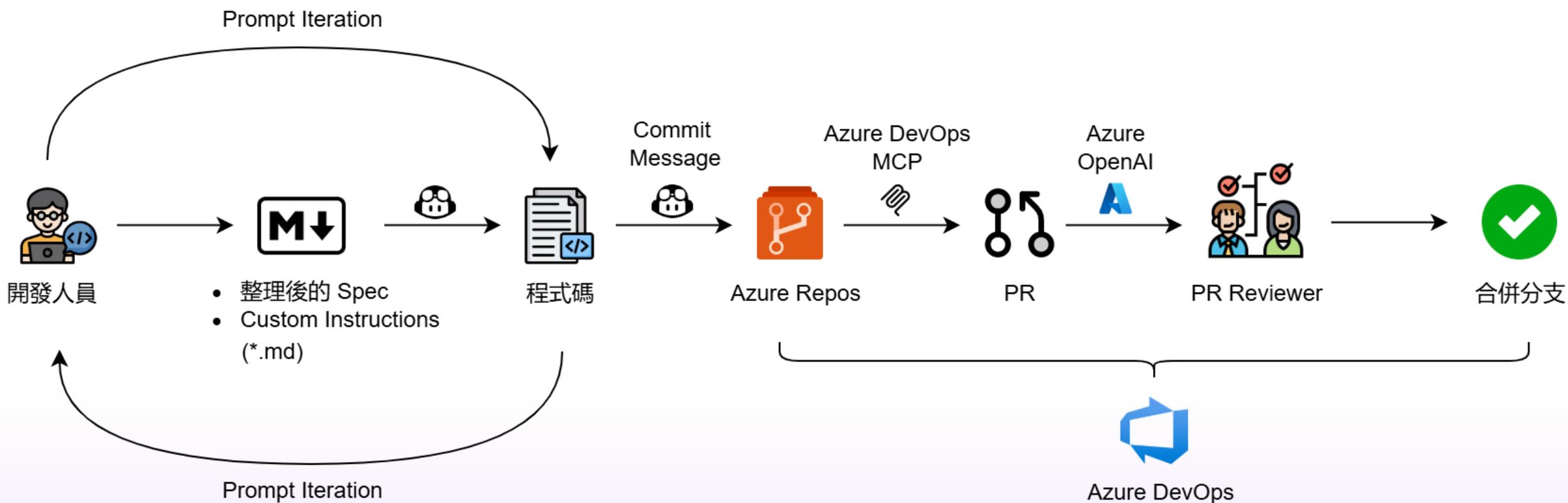
- 我可以立即替你確認頁面內容是否與本地 `api.md` 完全一致 (比對字元差異), 並把「Verify wiki page created」標為完成。是否要我執行?
- 或者我可以把這次變更建立成一個簡短變更紀錄 (PR / commit 註記, 若需要)。

你要我現在做「驗證內容並完成待辦」嗎?

🔄 ↶ 🗨️ 📄

GPT-5 mini • 0x

程式開發流程



Coding Guideline

The image shows a code editor window on the left and a chat interface on the right. The code editor displays a document titled 'csharp-coding-guideline.instructions.md' with the following content:

```
.github > instructions > csharp-coding-guideline.instructions.md > # C# 程式碼撰寫規範 > ## 程式碼格式與可讀性 (Formatting & Readability)
1 ---
2 description: 'C# 應用程式開發之程式碼撰寫規範'
3 applyTo: '**/*.cs'
4 ---
5
6 # C# 程式碼撰寫規範
7
8 ## 命名規範 (Naming Conventions)
9
10 一致的命名規範是程式碼可讀性的核心。建議遵循 Microsoft 的指導方針。
11
12 | 元素 | 命名規範 | 範例 |
13 |-----|-----|-----|
14 | 介面 | 前綴 'I' + PascalCase | `IAsyncRepository`, `ILogger` |
15 | 公開(public)成員 | 帕斯卡命名法 (PascalCase) | `public int MaxCount;`, `public void GetData()` |
16 | 參數、區域變數 | 小駝峰式命名 (camelCase) | `int userCount`, `string customerName` |
17 | 私有/內部欄位 | 底線(_) + 小駝峰式 | `private string _connectionString;` |
18 | 常數 (const) | 帕斯卡命名法 (PascalCase) | `public const int DefaultTimeout = 5000;` |
19 | 泛型型別參數 | 前綴 'T' + 描述性名稱 | `TKey`, `TValue`, `TResult` |
20 | 非同步方法 | 'Async' 後綴 | `GetUserAsync`, `DownloadFileAsync` |
21
22 ## 程式碼格式與可讀性 (Formatting & Readability)
23
24 一致的格式讓程式碼更容易視覺解析。
25
26 | 項目 | 規則 | 說明 |
27 |-----|-----|-----|
28 | 縮排 | 使用 4 個空白 | 請用 4 個空白而非 tab。cs 檔必須使用 4 個空白。 |
29 | 括號 | 總是使用大括號 {} | 控制語句 (如 if, for, while) 即使只有一行, 也一定要加大括號。 |
30 | 空行 | 邏輯分隔 | 方法定義、屬性定義、或邏輯分隔區塊之間應加空行。 |
31 | 陳述句撰寫 | 一行一陳述 | 每行只能寫一個陳述句。 |
32 | var 關鍵字 | 僅在型別明確時使用 | 僅當右側型別能清楚推斷時才用 var。 |
33 | 命名空間 | 使用檔案範圍命名空間 | C# 10 以上, 請使用檔案範圍命名空間減少不必要縮排。 |
34 | 註解 | 撰寫 XML 格式註解 | 為 class 和 function 撰寫 XML 格式註解。 |
35
36 每個 function 上方都要有 summary 註解, 說明此 function 的用途、參數與回傳值。
37
38 ## 語言功能通用 (Language Features)
39
40 善用最新 C# 功能, 讓程式碼更精簡與高效。
41
42 | 功能 | 說明 | 範例/參考 |
```

The chat interface on the right shows a conversation with a code review assistant. The assistant has analyzed the provided code and generated a list of todos for the files: TestRepository.cs, HomeController.cs, Program.cs, and test.cs. The assistant's response includes:

- A button to check if the attachments conform to the coding guideline.
- A list of files: TestRepository.cs, HomeController.cs, Program.cs, test.cs.
- A message: "Used 5 references" with a list of files: csharp-coding-guideline.instructions.md, csharp.instructions.md, dotnet-architecture-good-practices.instructions.md, dotnet-framework.instructions.md, and github.copilot.chat.codeGeneration.instructions.
- A message: "我先要建立工作清單, 然後逐步檢查每個檔案並彙總結果。接下來我會建立 TODO 清單並標記第一項為進行中。"
- A message: "Created 6 todos"
- A message: "進度更新: 已建立檢查清單並開始第一項 (範疇與規則確認)。下一步我會檢查 TestRepository.cs。"
- A message: "Starting (2/6) Review TestRepository.cs"
- A message: "接下來我會把 TestRepository.cs 檢查結果標記為完成, 並把 HomeController.cs 標記為進行中。之後會回報具體檢查要點和建議修正。"
- A message: "Starting (3/6) Review HomeController.cs"
- A message: "現在我會把剩下的 TODO 項目標記為完成, 表示我已完成對所有附件的審查並準備報告。接著會回報檢查要點與建議。"
- A message: "Completed (3/6) Review HomeController.cs"
- A message: "進度更新: 我已建立並完成 TODO 清單, 逐一檢查你附加的四個檔案, 現在彙整每個檔案是否符合專案 coding guideline (包含 DDD / SOLID / 命名/格式/例外/非同步/文件註解等), 並提出優先修正建議與範例修改。"
- A button to add context.
- A message: "Add context (#), extensions (@), commands (/)"
- A dropdown menu for the agent: Agent GPT-5 mini

Code Review & Refactor

`review-and-refactor.prompt.md` × `TestRecord.cs` `HomeController.cs` `TestRepository.cs` `Program.cs` CHAT

```
.github > prompts > review-and-refactor.prompt.md > ...
1 ---
2 mode: 'agent'
3 description: 'Review and refactor code in your project according to defined instructions'
4 ---
5
6 ## Role
7
8 You're a senior expert software engineer with extensive experience in maintaining projects over a long time and
9 ensuring clean code and best practices.
10
11 ## Task
12 1. Take a deep breath, and review all coding guidelines instructions in `.github/instructions/*.md` and `.github/
13 copilot-instructions.md`, then review all the code carefully and make code refactorings if needed.
14 2. The final code should be clean and maintainable while following the specified coding standards and
15 instructions.
16 3. Do not split up the code, keep the existing files intact.
17 4. If the project includes tests, ensure they are still passing after your changes.
18 所有回覆以繁體中文回答。
```

Follow instructions in `review-and-refactor.prompt.md`.

`HomeController.cs` `TestRepository.cs` `TestRecord.cs` `Program.cs`

Used 5 references

- `csharp-coding-guideline.instructions.md` `.github\instructions`
- `csharp.instructions.md` `.github\instructions`
- `dotnet-architecture-good-practices.instructions.md` `.github\instructions`
- `dotnet-framework.instructions.md` `.github\instructions`
- `github.copilot.chat.codeGeneration.instructions`

我將先建立工作計畫並標記第一個步驟為進行中，之後會依計畫修改四個檔案 (`Program.cs`、`TestRepository.cs`、`HomeController.cs`、`TestRecord.cs`)。

✓ Created 4 todos

下一步我將編輯 `Program.cs`、`TestRepository.cs`、`HomeController.cs`、`TestRecord.cs`，改善 DI 生命週期、加上 XML 註解、使用 `ConfigureAwait(false)`、更明確的例外以及參數驗證。進行一次性批次修改。

`Program.cs`

我要一次性修改三個檔案：`TestRepository.cs`、`HomeController.cs`、`TestRecord.cs` (包含把 `Program.cs` 的 DI lifetime 改為 `Scoped`)。接下來會套用重構：加入 XML 文件註解、參數檢查、更好的例外訊息、`ConfigureAwait(false)`，以及把 `repository` 註冊改為 `Scoped`。

`TestRepository.cs` +29 -4

接下來我會修正連線字串的 null 分配警告、將 DI 註冊改為 `Scoped`，並在 `HomeController.cs`、`TestRecord.cs` 加入 XML 註解與建議的 null 檢查。一次性套用三個檔案的變更。

> 4 files changed +65 -9 Keep Undo

Add Context...

輸入 / 選擇 Custom Prompt

Agent GPT-5 mini

Branch Policy – Build Validation

The screenshot displays a GitHub pull request interface for a repository named 'SeRMS'. The pull request is titled 'Bug fix' and is in a 'Completed' state. It shows that the pull request was merged into the 'master' branch. The interface includes a navigation bar with 'Overview', 'Files', 'Updates', 'Commits', and 'Conflicts'. A sidebar on the left contains various GitHub navigation icons. The main content area shows a list of checks that passed, including 'Required check succeeded', 'SeRMS-PR Reviewer Build succeeded', 'Required reviewers have approved', and 'No merge conflicts'. Below the checks is a 'Description' section with a bulleted list of changes. On the right side, there are sections for 'Reviewers' (showing 'Tim Lai' as an approved reviewer), 'Tags' (showing 'No tags'), and 'Work items' (listing several bugs with their titles and update dates).

SeRMS / Repos / Pull requests / SeRMS

Search

Delete source branch

Bug fix

Completed 1103 [Avatar] proposes to merge SRM into master 0/8 comments resolved

Overview Files Updates Commits Conflicts

Tim Lai completed this pull request Sep 1 Cherry-pick Revert

Merged PR 103: Bug fix
1c8c1f2c Sep 1 at 4:41 PM

Show details

- Required check succeeded
- SeRMS-PR Reviewer Build succeeded
- Required reviewers have approved
- No merge conflicts
Last checked Sep 1

Description

- fix(SRM/Airport): 優化分數計算邏輯, 延遲執行以確保所有屬性已設置 [671 Airport - 分數顯示異常](#) Done
- fix(SRM/EnrouteCheck): 起訖航班輸入框添加自動完成事件處理
[674 EnrouteCheck - Flight Sector的起訖航班輸入框輸入後需自動提示Airport的清單供選擇](#) Done
- feat(SYS/User): 增加最後異動時間顯示
- fix(SYS/Role): 調整每頁顯示數量從10改為20 [672 帳號權限管理 - 角色管理查詢每頁顯示20筆](#) Done
- fix(SYS/RoleFunction): 修正JSTree節點取消勾選邏輯
[673 帳號權限管理 - 角色授權管理於 Step 1.功能維護勾選單一功能再取消, Step 2. 功能授權維護會殘留階層](#) Done

Reviewers

Required

- Tim Lai Approved

Optional

- Approved

Tags

No tags

Work items

- Bug 670: 帳號權限管理 - 帳號管理所 Updated Tuesday, Done
- Bug 671: Airport - 分數顯示異常 Updated Tuesday, Done
- Bug 672: 帳號權限管理 - 角色管理查詢 Updated Tuesday, Done
- Bug 673: 帳號權限管理 - 角色授權管理 Updated Tuesday, Done
- Bug 674: EnrouteCheck - Flight Sector Updated Tuesday, Done

Branch Policy – Build Validation

The screenshot displays a GitHub pull request for a "Bug fix" in the "SeRMS" repository. The pull request is completed and has been merged into the "master" branch. The interface shows various status checks and a list of work items.

Overview: Completed 1103 [Avatar] proposes to merge SRM into master 0/8 comments resolved

Check Status:

- Required check succeeded
- SeRMS-PR Reviewer Build succeeded** (highlighted with a red box)
- Required reviewers have approved
- No merge conflicts (Last checked Sep 1)

Reviewers:

- Required: Tim Lai (Approved)
- Optional: [Avatar] (Approved)

Work items:

- Bug 670: 帳號權限管理 - 帳號管理所層 Updated Tuesday, Done
- Bug 671: Airport - 分數顯示異常 Updated Tuesday, Done
- Bug 672: 帳號權限管理 - 角色管理查詢 Updated Tuesday, Done
- Bug 673: 帳號權限管理 - 角色授權管理 Updated Tuesday, Done
- Bug 674: EnrouteCheck - Flight Sector Updated Tuesday, Done

Description:

- fix(SRM/Airport): 優化分數計算邏輯, 延遲執行以確保所有屬性已設置 | 671 Airport - 分數顯示異常 | Done
- fix(SRM/EnrouteCheck): 起訖航班輸入框添加自動完成事件處理 | 674 EnrouteCheck - Flight Sector的起訖航班輸入框輸入後需自動提示Airport的清單供選擇 | Done
- feat(SYS/User): 增加最後異動時間顯示
- fix(SYS/Role): 調整每頁顯示數量從10改為20 | 672 帳號權限管理 - 角色管理查詢每頁顯示20筆 | Done
- fix(SYS/RoleFunction): 修正JSTree節點取消勾選邏輯 | 673 帳號權限管理 - 角色授權管理於 Step 1.功能維護勾選單一功能再取消, Step 2. 功能授權維護會殘留階層 | Done

Branch Policy – Build Validation

The screenshot shows a GitHub pull request interface. The breadcrumb navigation at the top reads: SeRMS / Repos / Pull requests / SeRMS. The search bar is empty. The left sidebar contains navigation icons for Overview, Files, Updates, Commits, and Conflicts. The main content area displays three files with associated build validation comments from the 'SeRMS Build Service (M118)'.

RiskCityBranchController.cs (SeRMS.WebAPI/Controllers/SRM/RiskCityBranchController.cs) - Sep 1

- 新增的 `ReportSubject` 行中使用了多個三元運算符，可能會使代碼的可讀性降低。建議將邏輯拆分成更具體的方法或使用更清晰的條件語句來提高可讀性。
- 在生成 URL 字符串時，未對 `id` 進行任何格式或安全檢查。確保 `id` 的值是安全的，並且不會導致 URL 攻擊（例如，SQL 注入或跨站腳本攻擊）。
- 在 `ReportSubject` 和 `Subject` 中使用字符串拼接時，請確保 `subject` 的值已被驗證和清理，以防止任何可能的安全漏洞，例如跨站腳本攻擊。

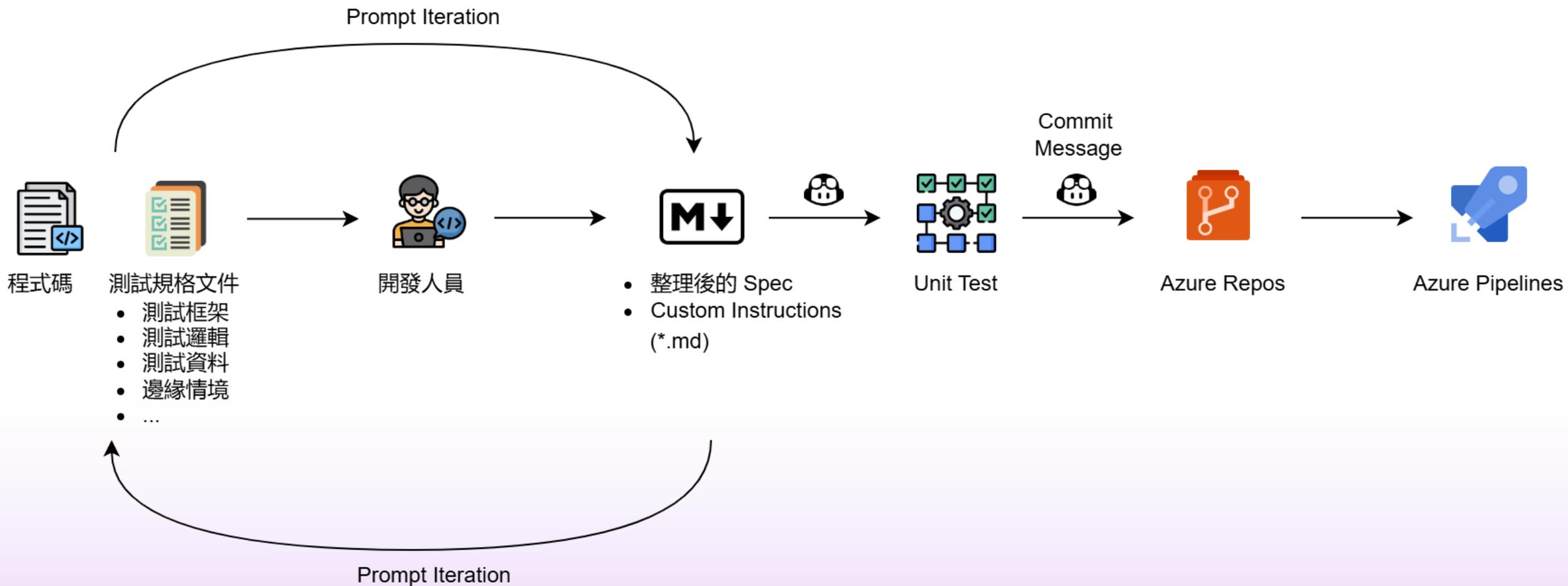
Index.cshtml (SeRMS.Web/Areas/SRM/Views/RiskChecklist/Index.cshtml) - Sep 1

- 在 `typeChangeQuery` 方法中，當 `vm.isQuery` 被設置為 `true` 時，可能需要考慮是否有必要在方法內部進行額外的狀態檢查或處理，以確保狀態的正確性。
- 在 `GetCityScoreClass` 和 `GetAirportScoreClass` 方法中，對於分數的分類邏輯，可能需要考慮是否有必要將分數範圍的界限值設置為常量，以提高可讀性和可維護性。
- 在 `GetAirportScoreClass` 方法中，使用 `bg-ext-danger` 作為 CSS 類名可能需要確認該類名是否存在於樣式表中，以避免樣式錯誤。
- 在使用 `moment` 格式化日期時，可能需要考慮是否有必要檢查 `moment` 是否已加載，以避免在未加載時出現錯誤。
- 在 `SendMail` 方法中，可能需要考慮添加錯誤處理機制，以確保在郵件發送失敗時能夠正確處理錯誤。

Detail.cshtml (SeRMS.Web/Areas/SRM/Views/RiskCity/Detail.cshtml) - Sep 1

- 在新增的 JavaScript 代碼中，使用 `window.location.hash` 來獲取 URL 的 hash 值並自動切換頁籤。雖然這是一個常見的做法，但請確保 URL hash 值的來源是安全的，避免可能的 XSS 攻擊。如果 hash 值是由用戶輸入或可控的，建議對 hash 值進行驗證或清理。
- 建議在文件末尾添加一個換行符，以符合 POSIX 標準，這有助於在某些工具或系統中避免潛在的問題。

Unit Test



Unit Test - NUnit

The image shows a VS Code editor with a chat window on the right. The chat window contains a prompt and a response. The prompt is a YAML file named `csharp-nunit.prompt.md` with the following content:

```
.github > prompts > csharp-nunit.prompt.md > ...
1 ---
2 mode: 'agent'
3 Configure Tools...
4 tools: ['changes', 'codebase', 'editFiles', 'problems', 'search']
5 description: 'Get best practices for NUnit unit testing, including data-driven tests'
6 ---
7 # NUnit Best Practices
8
9 Your goal is to help me write effective unit tests with NUnit, covering both standard and data-driven testing
10 approaches.
11
12 ## Project Setup
13 - Use a separate test project with naming convention `[ProjectName].Tests`
14 - Reference Microsoft.NET.Test.Sdk, NUnit, and NUnit3TestAdapter packages
15 - Create test classes that match the classes being tested (e.g., `CalculatorTests` for `Calculator`)
16 - Use .NET SDK test commands: `dotnet test` for running tests
17
18 ## Test Structure
19
20 - Apply `[TestFixture]` attribute to test classes
21 - Use `[Test]` attribute for test methods
22 - Follow the Arrange-Act-Assert (AAA) pattern
23 - Name tests using the pattern `MethodName_Scenario_ExpectedBehavior`
24 - Use `[SetUp]` and `[TearDown]` for per-test setup and teardown
25 - Use `[OneTimeSetUp]` and `[OneTimeTearDown]` for per-class setup and teardown
26 - Use `[SetUpFixture]` for assembly-level setup and teardown
27
28 ## Standard Tests
29
30 - Keep tests focused on a single behavior
31 - Avoid testing multiple behaviors in one test method
32 - Use clear assertions that express intent
33 - Include only the assertions needed to verify the test case
34 - Make tests independent and idempotent (can run in any order)
35 - Avoid test interdependencies
36
37 ## Data-Driven Tests
38
39 - Use `[TestCase]` for inline test data
40 - Use `[TestCaseSource]` for programmatically generated test data
```

The chat window on the right shows the following response:

Follow instructions in `csharp-nunit.prompt.md`.
CalculatorApp 專案

Used 1 reference

我要先讀取指定的 prompt 指示檔案，然後檢查 `CalculatorApp` 專案內容以執行要求。接下來我會開啟那個指示檔案。

Read `csharp-nunit.prompt.md`

接下來我會檢查 `CalculatorApp` 與 `CalculatorApp.Tests` 專案檔案與原始碼以評估需要的變更。

Created 5 todos

我將先閱讀專案檔案與相關原始碼：`Calculator.cs`，`CalculatorApp.csproj`，`CalculatorApp.Tests/CalculatorApp.Tests.csproj`。接下來我會開啟這些檔案。

Read `Calculator.cs`

Read `CalculatorApp.csproj`

接下來我會列出專案資料夾以確認 `CalculatorApp.Tests` 是否存在並查看內容。這會列出 `CalculatorApp.Tests` 目錄下的檔案。

Read `CalculatorApp.Tests`

我要建立測試專案與測試類別檔案，並加入 NUnit 套件與專案參考，然後執行測試以驗證。接下來我會新增 `CalculatorApp.Tests` 專案與 `CalculatorTests`。

`CalculatorApp.Tests.csproj`

我將新增 `CalculatorTests.cs` 使用 NUnit，上面會包含標準測試與 data-driven 範例 (`[TestCase]`)。接下來我會新增測試檔案。

`CalculatorTests.cs`

Add Context...
輸入 / 選擇 Custom Prompt

Agent GPT-5 mini

Unit Test - NUnit

The image shows a Visual Studio Code editor with a C# file named `CalculatorTests.cs` open. The code defines several test methods for a calculator application. The `Add_PositiveNumbers_ReturnsSum()` method is highlighted with a yellow box, showing three `[TestCase]` attributes: `[TestCase(5, 2, 3)]`, `[TestCase(-1, -3, 2)]`, and `[TestCase(0, 0, 0)]`. Other methods include `Add_TestCases_ResultMatches`, `Subtract_TestCases_ResultMatches`, `Multiply_TestCases_ResultMatches`, and `Divide_ByNonZero_ReturnsQuotient`. Each method uses `Assert.That` to verify the results.

On the right side, there is a chat window titled "CHAT" showing a conversation with an AI assistant. The chat content includes:

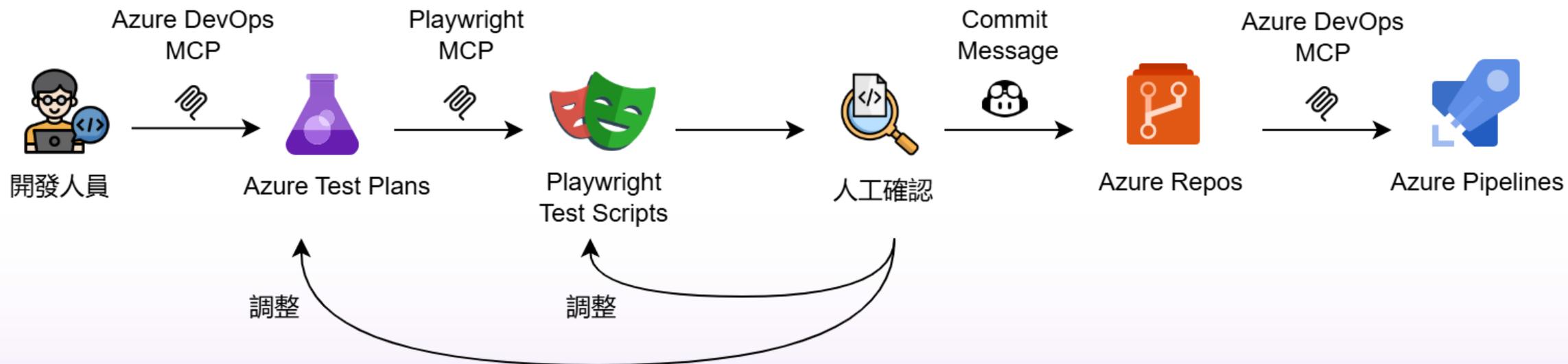
- A prompt: `2025\CalculatorApp.Tests'; dotnet test -v normal?`
- A response: "This will continue to poll for output to determine when the terminal becomes idle for up to 2 minutes."
- A user message: "我要修正錯誤的 TestCase 期待值，然後重新執行 `dotnet test` 驗證修正。接下來先更新測試欄。"
- A code block: `cd 'd:\SystemSoftware\Work\DevDays 2025\CalculatorApp.Tests'; dotnet test -v minimal`
- A response: "我已完成測試、修正並驗證所有測試通過，下一步會更新待辦清單標記已完成的項目，並向您回報修改內容與如何執行測試。"
- A status message: "Starting (5/5) 回報與下一步建議"
- A user message: "我要回報我對 `CalculatorApp` 專案所做的變更，並提供如何執行與接下來的建議。"
- A response: "已完成 (重點):"
- A list of updates:
 - 新增測試專案: `CalculatorApp.Tests.csproj`
 - 已加入套件引用: `Microsoft.NET.Test.Sdk`、`NUnit`、`NUnit3TestAdapter`
 - 以 `net9.0` 為目標框架，並參考 `CalculatorApp.csproj`
 - 新增測試類別: `CalculatorTests.cs`
 - 使用 `[TestFixture]`、`[SetUp]`，並遵循 Arrange-Act-Assert 模式
 - 包含標準測試與 data-driven 範例 (使用 `[TestCase]`)
 - 測試覆蓋 `Add`、`Subtract`、`Multiply`、`Divide` (含除以零例外檢查)
 - 執行測試並修正: 我執行 `dotnet test` 並修正了一個錯誤的預期值 (`Subtract` 的 `TestCase`)，目前所有測試均通過 (10 個測試, 0 失敗)。

At the bottom of the chat window, there is an input field with "Add Context...", a dropdown menu showing "CalculatorTests.cs", and a "GPT-5 mini" model selector.

End to End Test



VSCode



Azure Test Plans – Test Case

TEST CASE 322

322 KM / Country - Query 查詢列表

0 Comments Add Tag

State Design Area SeRMS\SeRMS Team Reason New Iteration SeRMS\Sprint 8

Steps

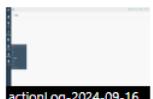
Steps	Action	Expected result	Attachments
1.	開啟 SeRMS 系統		
2.	進入 KM / Country 功能		
3.	Click on button 'Query'	顯示所有查詢結果	
4.	Update input field "Country Code" with value 'T'		
5.	Click on button "Query"	顯示所有 Country Code 包含 T 的資料	
6.	Click on text box "Country Name"		
7.	Update input field with value 'AI'		
8.	Click on button "Query"	依據目前資料，再篩選 Country Name 包含 "AI" 的資料	
9.	Click on dropdownlist "Country Type"		
10.	Click on "有飛"		
11.	Click on button "Query"	查無任何資料	
12.	Click on button "Clear"	清除所有輸入欄位	
13.	Click on button "Query"	顯示所有資料	

Click or type here to add a step



322*: KM / Country - Query 查詢列表

- 開啟 SeRMS 系統
- 開啟網址 <https://dlar-sit.japaneast.cloudapp.azure.com/SeRMS>
EXPECTED RESULT
顯示系統首頁
- 進入 KM / Country 功能
- Click on 左側選單 "Knowledge Management"
EXPECTED RESULT
出現選單

- Click on link "Country"
EXPECTED RESULT
導頁至 Country 功能

- Click on button 'Query'
EXPECTED RESULT
顯示所有查詢結果
- Update input field "Country Code" with value 'T'
- Click on button "Query"
EXPECTED RESULT
顯示所有 Country Code 包含 T 的資料
- Click on text box "Country Name"
- Update input field with value 'AI'
- Click on button "Query"
EXPECTED RESULT
依據目前資料，再篩選 Country Name 包含 "AI" 的資料
- Click on dropdownlist "Country Type"
COMMENT
沒有顯示這個下拉選單
- Click on "有飛"

機場 OA 配置

取得 Azure Test Plans 相關資訊

TestCase.prompt.md X

.github > prompts > TestCase.prompt.md

```
1 ---
2 mode: agent
  Configure Tools...
3 tools: ['editFiles', 'codebase', 'problems', 'changes', 'azure-devops']
4 description: 取得 Azure DevOps 測試案例的詳細資訊
5 ---
6 Get me the details of the test cases (do not action anything yet, just give me the details of each test case).
7 Test Information:
8
9 * ADO Organization: Org_Name
10 * Project: Project_Name
11 * Test Plan ID: Test_Plan_ID
12 * Test Suite ID: Test Suite ID
```

依據 Test Plan 產生 Playwright 程式碼

```
Playwright.prompt.md X
.github > prompts > Playwright.prompt.md > # Project Context
1 ---
2 mode: agent
  Configure Tools...
3 tools: ['editFiles', 'codebase', 'problems', 'changes', 'context7', 'azure-devops', 'playwright']
4 description: 依據 Azure DevOps 測試案例內容，協助撰寫 Playwright 測試腳本
5 ---
6 Imagine you are an experienced Software Engineer helping me write high-quality Playwright test scripts in TypeScript based on the test cases I provided. Please go over the task twice to
  make sure the scripts are accurate and reliable. Avoid making things up and do no hallucinate. Use all the extra information outlined below, to write the best possible scripts, tailored
  for my project.
7
8 # Project Context 專案內容
9 Look at the "Project_name" folder, to get more insights (if your project is quite large, use the below section to be more concrete and reference specific folders/files).
10
11 My project structure includes:
12
13 * Authentication helpers:    /*Add/folder/path*
14 * Existing sample tests:    /*Add/folder/path*
15 * Playwright config:        /*Add/folder/path*
16 * Test Structure:           /*Add/folder/path/test-1656280.spec.ts*
17 * The project's UX components are in the following folder: /*Add/folder/path*.
18
19 # Test Structure Requirements 測試結構
20 For each test, please follow this structure:
21 1. Clear test description using *test.describe()* blocks
22 2. Proper authentication setup before any page navigation
23 3. Robust selector strategies with multiple fallbacks
24 4. Detailed logging for debugging
25 5. Screenshot captures at key points for verification
26 6. Proper error handling with clear error messages
27 7. Appropriate timeouts and wait strategies
28 8. Verification/assertion steps that match the test case acceptance criteria
29
30 # Robustness Requirements 穩健性
31 Each test should include:
32 1. Retry mechanisms for flaky UI elements
33 2. Multiple selector strategies to find elements
34 3. Explicit waits for network idle and page load states
35 4. Clear logging of each test step
36 5. Detailed error reporting and screenshots on failure
37 6. Handling of unexpected dialogs or notifications
38 7. Timeout handling with clear error messages
```

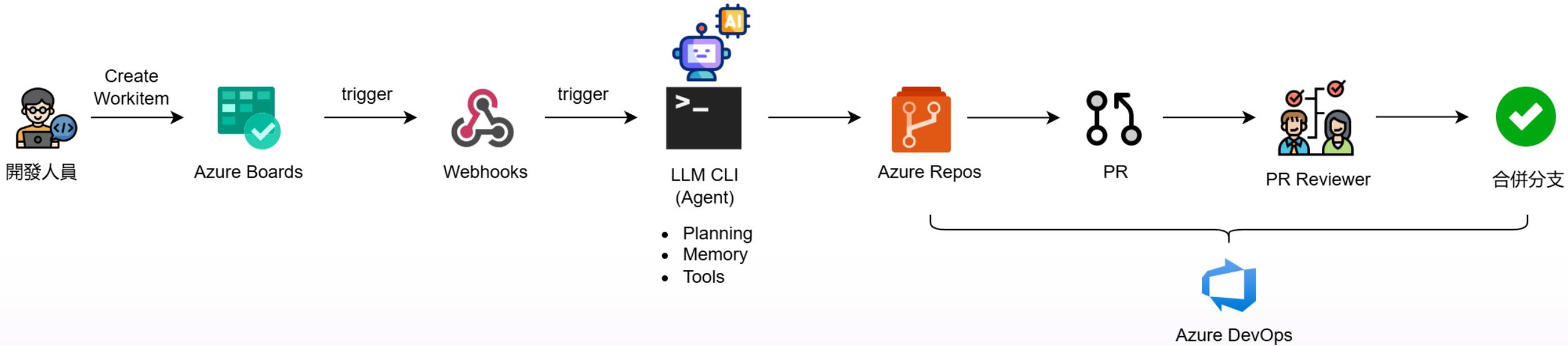
依據 Test Plan 生成 Playwright 程式碼

Playwright.prompt.md

.github > prompts > Playwright.prompt.md > # Project Context

```
39
40 # Environmental Considerations 環境設定
41 The tests will run in:
42 * CI/CD pipeline environments
43 * Headless mode by default
44 * Potentially with network latency
45 * Different viewport sizes
46
47 # Example Usage 範例使用
48 Please provide a complete implementation with:
49 1. Helper functions for authentication and common operations
50 2. Full test implementation for each test case
51 3. Comments explaining complex logic
52 4. Guidance on test execution
53
54 # Authentication Approach 驗證方法
55 In order for the tests to be executed, we need to authenticate the application. Use the below auth approach:
56 //{you need to define the authentication steps - if this is already defined for your project, instruct Copilot how to use it. If your scenarios do not require auth, you can remove this
57 part from the prompt.}
58
59 # Configuration Reference 參數設定
60 For timeouts, screenshot settings, and other configuration options, please refer to:
61 //{Add a reference to a specific file, etc. for better context}
62 I want these tests to be maintainable, reliable, and provide clear feedback when they fail.
```

進一步與 LLM CLI 整合



落地實踐注意事項

1

資料隱私

確保敏感資料的安全，避免直接輸入到 AI 模型中。



2

審核產出結果

檢視 AI 生成的程式碼與文件，以確保正確性與安全性。



3

上下文精確性

維護最新且完整的參考資料，以獲得準確的 AI 結果。



4

規格一致性

自訂提示指令以符合團隊開發規範，減少理解上的誤差。



5

持續學習

學習新技能與最佳實踐，善用自動化提升效能，主動管理導入風險。



Summary



AI 正在推動 DevOps 團隊工作從「**被動**」走向「**主動**」、「**智慧**」與「**自動化**」。



藉由 GitHub Copilot 及 Azure DevOps MCP 等工具，團隊能**統整上下文**、**加速任務分解與規格驅動開發**，提高協作效率。



持續最佳化 Prompt 與上下文整合，是團隊獲取 AI 助力效能的關鍵。



開放探索新工具與客製流程，能讓團隊**快速適應未來 AI 驅動的開發與協作新時代**。

精誠軟體服務致力於提供客戶全面性資訊科技應用解決方案，
配合客戶運營發展，規劃多元且彈性的部署策略協助企業全力邁向數位轉型之路，推動營運前行。
以科技的力量安定人心，以數位韌性推動產業一起向前行。

精誠軟體服務 台北/新竹/台中/高雄

客服信箱：digitalmk@systemxsoftware.com.tw



Website



Facebook



YouTube



LINE



Instagram

Thank You

聯絡我們

