旅遊導向智慧 LINE BOT

Group 3

1113353 1121507 1121508

1123301 1123306

OVERVIEW

分工表

1113353	主題發想、匯出資料庫、MCP TOOL、測試、海報
1121507	主題發想、翻譯、測試、海報
1121508	主題發想、情緒分析、翻譯、測試、海報
1123301	主題發想、ADK、MCP TOOL、測試、資料庫處理、分錢系統、海報
1123306	主題發想、爬蟲、測試、整合、分錢系統、海報、資料庫架設

主題簡介

旅遊導向智慧 Line Bot

- 整合多模態輸入（文字 / 語音 / 圖片）與雲端 AI 服務
- 提供即時翻譯、景點推薦、語意理解及分錢功能
- 以 Python Flask 為後端核心，結合語言與語音服務及 Google Gemini

功能描述

多模態翻譯

- 文字輸入→Translator service
- 語音輸入(Genimi辨識)
- 圖片輸入(Genimi辨識)

功能描述

美食推薦

- 依 Line Location Message 推薦附近餐廳
- 從使用者文字擷取關鍵字進行推薦
- 使用爬蟲抓取Google Map 的資訊並存到資料庫

功能描述

語言智慧模組

- 關鍵字記憶
- 情緒分析
- 對話生成與語意整合

功能描述

記帳系統

- 建立共同群組並且可共同記帳
- 可隨時列出結至當下的總花費
- 一鍵結算所有人的花費，並要給誰多少錢

實作細節

Line SDK

我們將所有的決策跟執行都交給ADK，所以Line的SDK，只是接收資訊跟回傳結果的工具。接受資訊後，透過 call_adk 的 function 將輸入的資訊丟給adk

實作細節

ADK

ADK是我們整個系統的大腦，基本所有輸入的資訊都是交給ADK + Gemini 負責，包含所有工具的執行及答覆。

實作細節

Sessions

我們遇到的第一個問題即是，要如何突破15分鐘的session限制，我們使用資料庫來記憶使用者的session，讓對話都是連續且有記憶的

Audio & Image

另一部分，因為ADK雖然是用Gemini來做，但他卻沒法直接傳圖片或音檔供它辨識，為此我們使用base64將圖片格式化，再讓agent辨識。

實作細節

匯出資料庫

這個部分BOT需要從 MySQL 匯出 foodDb，產生 users.xlsx，並把匯出的檔案回傳給使用者下載，讓我們在聊天室窗裡即可完成資料匯出。

實作細節

匯出資料庫2

LINE 不提供『檔案訊息』，所以我們的解法是把檔案放到伺服器上，回一個下載連結給使用者。所以我們決定聊天入口 → Agent 決策 → Tool 執行 → 把結果包回 UI 的流程。

實作細節

MCP TOOL

使用者提到「匯出資料庫」→ 呼叫 db_export tool，然後在呼叫匯出工具後 只輸出 tool 回傳的 JSON，並讓 webhook 端可以穩定判斷這是不是匯出結果

實作細節

HTTP 下載流程

我們把檔案下載變成一個標準 HTTP 下載流程，LINE 只負責讓使用者開啟網址。此外後端還用 token 對應檔案網址加密，避免路徑被猜到而被亂下載檔案。

實作細節

爬蟲

我們在推薦美食的部分，使用爬蟲來擴充資料庫。

較易上手)來爬取Google Map上的資訊。

實作細節

記帳分錢系統

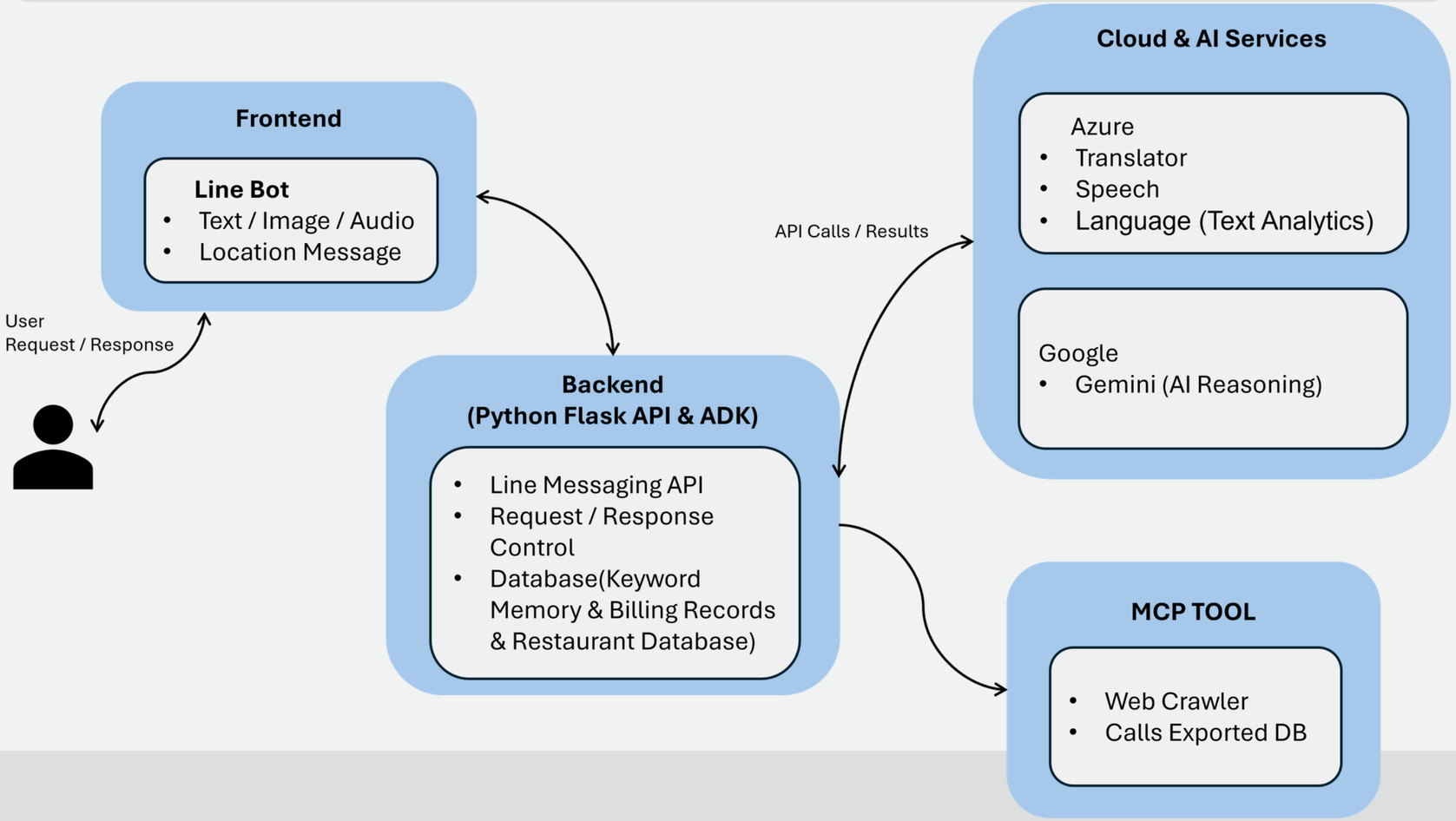
我們使用Liff 及 資料庫 實作出分帳的功能，
我們把資料庫會用到的CRUD功能都分別先弄出來。
再透過Flask 處理邏輯架構

實作細節

資料庫設計

使用關聯式資料庫建立結構化資料表，透過外鍵關聯與 JOIN 做查詢（例如查群組成員、查群組記帳），用 transaction 保證多步驟操作的一致性（加入群組、寫入記帳等）

系統架構


Frontend
Line Bot
• Text / Image / Audio
• Location Message
User
Request / Response
Backend
(Python Flask API & ADK)
• Line Messaging API
• Request / Response Control
• Database(Keyword Memory & Billing Records & Restaurant Database)
API Calls / Results
Cloud & AI Services
Azure
• Translator
• Speech
• Language (Text Analytics)
Google
• Gemini (AI Reasoning)
MCP TOOL
• Web Crawler
• Calls Exported DB

問題與解決方案分享

路徑處理問題

在不同電腦執行的時候使用絕對路徑，在電腦切換時，常因路徑不一致，導致無法讀資料表或爬蟲。

SQL 指令錯誤排除

手敲 SQL 指令（如 INSERT 或 UPDATE），在輸入值得時候容易因為格式，易出錯且難以維護。

資料庫格式定義

使用SQL的json格式，容易出現不合格的格式，因此我們是推入TEXT，並且由後端統一管理(split，或是json.loads())

AI 回應的「幻覺」與精準度問題

大語言模型容易產生錯誤，不會照我們預期的設定跟prompt執行。

問題與解決方案分享

團隊協作與版本控制

在開發過程中，由於多人同時修改分錢邏輯、爬蟲模組與 MCP 串接，經常發生程式碼互相覆蓋或整合失敗的情況，導致開發進度受阻。
透過結構化 GitHub 分支管理策略 (Branching Strategy)、功能分支化 (Feature Branching)：為每個功能建立獨立分支，確保開發環境互不干擾。Pull Request：在合併回 main 主分支前進行程式碼審核，確保程式碼品質並提早發現 Bug。減少衝突：透過模組化設計與頻繁的同步更新，大幅降低了合併時的衝突機率，提升開發效率。

問題與解決方案分享

多模態整合的複雜度過高

文字、語音、圖片來自不同來源，每種輸入皆需不同的前處理流程，若未妥善設計，容易導致流程混亂與除錯困難。
因此我們採用 LINE Webhook Message Type 分流設計，依訊息類型將處理流程模組化，確保各功能彼此獨立且清楚。
透過 LINE SDK 提供的 decorator 機制，將不同訊息導向對應的 handler，並在完成必要的資料轉換（base64 編碼）後，統一轉為標準化的語意輸入格式交由 ADK 處理，使 ADK 作為系統的單一決策中樞，使整體架構更單純且容易維護。

問題與解決方案分享

Session 記憶導致 API 使用量快速增加

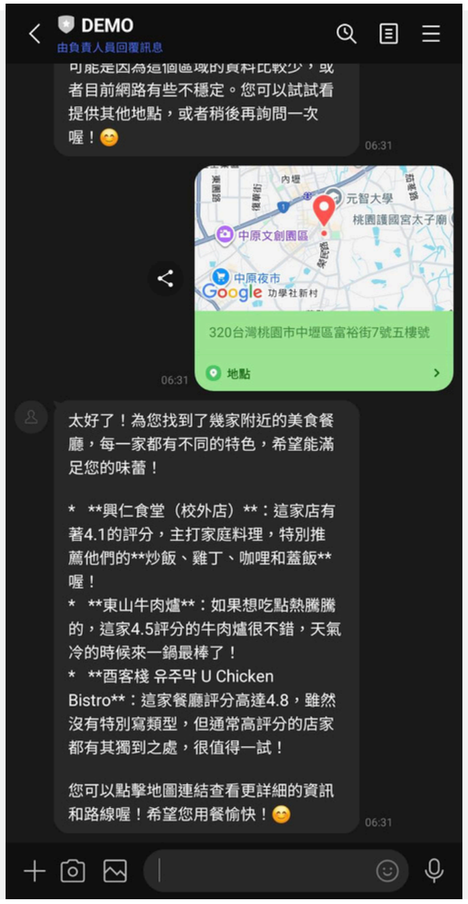
為維持對話記憶，我們最初將完整 session 傳入 Gemini，後來發現 session 越來越長會讓 token 扣得很快導致 token 使用量快速成長。
為避免 session 過度累積造成 token 消耗過快，我們對對話記憶設定有效期限，定期清除舊資料，只保留必要的上下文供 AI 使用。

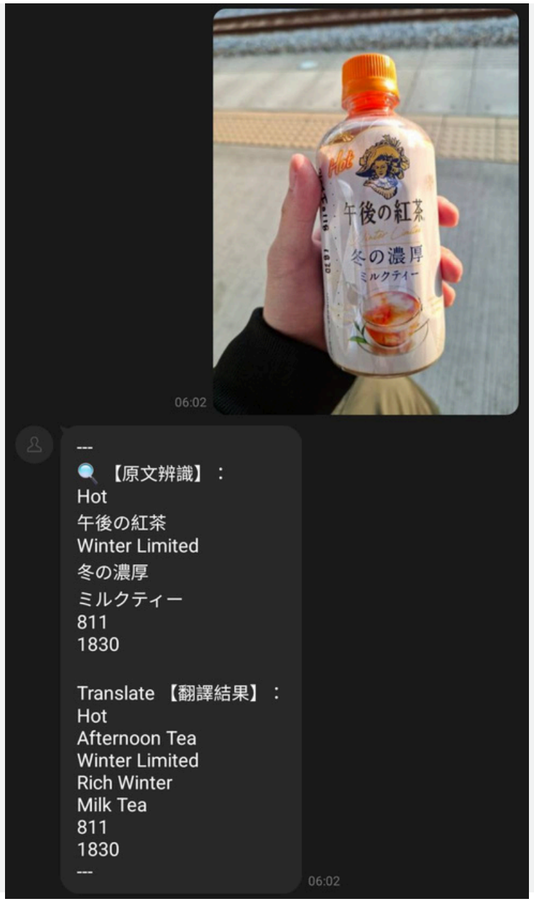
未來展望/心得

- 導入使用者偏好模型以提升推薦精準度
- 整合即時天氣、交通與路線規劃 API
- 優化分錢系統(不同消費金額、備註、line login)

- 多模態輸入的價值： 單純文字輸入已不足以滿足現代旅遊需求。透過開發圖片翻譯與語音處理，降低使用者輸入門檻。
- 從需求出發的實踐： 原本只專注於導覽，但加入「分錢』功能後，Bot 真正解決了旅遊結束後最尷尬的帳務問題，解決生活中的瑣碎痛點。

DEMO

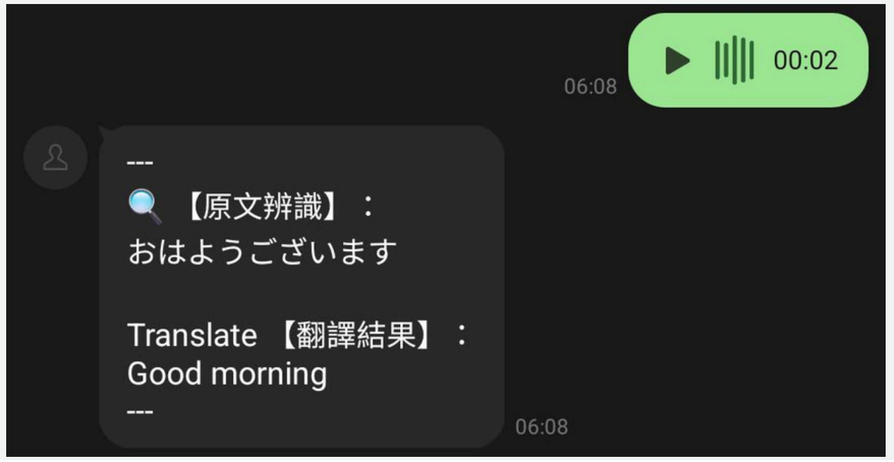

DEMO
由負責人員回覆訊息
可能是因為這個區域的資料比較少，或者目前網路有些不穩定。您可以試試看提供其他地點，或者稍後再詢問一次喔！😊
06:31
元智大學
中原文創園區
中原夜市
320台灣桃園市中壢區富裕街7號五樓號
地點
06:31
太好了！為您找到了幾家附近的美食餐廳，每一家都有不同的特色，希望能滿足您的味蕾！
* **興仁食堂（校外店）**：這家店有著4.1的評分，主打家庭料理，特別推薦他們的**炒飯、雞丁、咖哩和蓋飯**喔！
* **東山牛肉爐**：如果想吃點熱騰騰的，這家4.5評分的牛肉爐很不錯，天氣冷的時候來一鍋最棒了！
* **酉客棧 유주막 U Chicken Bistro**：這家餐廳評分高達4.8，雖然沒有特別寫類型，但通常高評分的店家都有其獨到之處，很值得一試！
您可以點擊地圖連結查看更詳細的資訊和路線喔！希望您用餐愉快！😊
06:31


06:02

🔍 【原文辨識】：
Hot
午後の紅茶
Winter Limited
冬の濃厚
ミルクティー
811
1830
Translate 【翻譯結果】：
Hot
Afternoon Tea
Winter Limited
Rich Winter
Milk Tea
811
1830

06:02


【岩壁旅行】帳單已結清！💸
還款建議：
• 逢 ➡ Bradon: $300
請成員們確認並完成轉帳。
22:46


06:08
00:02

🔍 【原文辨識】：
おはようございます
Translate 【翻譯結果】：
Good morning

06:08

THANK YOU