期中報告提案書

LLM 驅動的個人資訊管理系統

指導教授：

陳俊豪

組員：

C112110156 邱柏愷

C112110244 李承育

C112151148 徐慶峰

C112156233 蘇泓叡

1. 簡介與動機

1.1 專案目標

開發一個基於 Telegram 的智能個人助理機器人，探索對話式 AI 作為 Personal Information Management (PIM) 系統的新交互範式。透過 LLM (Large Language Model) 驅動的自然語言理解，讓使用者以自然語言管理個人資訊，包括記帳、待辦事項、行事曆、提醒與備忘錄等功能。

1.2 研究背景

1.2.1 從個人使用經驗看 PIM 的挑戰

想像這個場景：你在便利商店買了咖啡和三明治，花了 120 元。結帳後走出店門，你想記帳。但此時你要趕去搭捷運，手上還拿著咖啡。你需要：掏出手機、解鎖、找到記帳 App、等待載入、點選「新增支出」、在一長串類別中滑動尋找「飲食」或「早餐」、輸入金額 120、確認日期、可能還要選擇付款方式、最後點選儲存。

整個過程需要 30 秒以上和至少 6-7 次點擊。如果你正趕時間，你會想「等等再記」。然後到了晚上，你已經完全忘記這筆支出了。

更麻煩的是分類問題。「咖啡和三明治」應該算「早餐」還是「飲料」？如果是跟同事一起買的，算「社交」嗎？每次記帳都要做這種決定，久而久之就會覺得「算了，不記了」。

1.2.2 待辦事項的矛盾：工具比任務本身還複雜

待辦事項管理也有類似問題。很多人嘗試過 Todoist、Things、Microsoft To Do、Google Tasks 等各種工具，但最後都回到在 Line 開一個 1 人群組來記錄。

當想到「下週五前要完成期末報告」這件事時，在腦中只是一句話。但要把它記到待辦 App 裡，我需要：

1. 打開 App
2. 點選「新增待辦」

3. 輸入標題「完成期末報告」
4. 點開日期選擇器
5. 在日曆中找到「下週五」是哪一天
6. 決定要不要設提醒（要的話又是一連串設定）
7. 選擇優先級（高？中？低？）
8. 選擇專案分類（學業？個人？）
9. 點選儲存

這個流程讓人覺得記錄待辦事項的複雜度，甚至超過完成待辦事項本身。結果就是，寧願用最簡單的方式記在備忘錄裡，寫一句「下週五前：期末報告」就好了。

1.2.3 資訊碎片化：資料散落各處

更根本的問題是資訊碎片化。個人資訊散落在不同的地方：

- 記帳在 Money Manager
- 待辦在 Apple 提醒事項
- 行事曆在 Google Calendar
- 備忘錄在 Apple Notes
- 重要日期又記在手機的內建行事曆
- 臨時想法隨手寫在 Telegram 的「儲存的訊息」

當想回顧「上個月花了多少錢」「本週有哪些待辦」「之前記的那個停車位號碼」時，需要在多個 App 之間切換，每個 App 的介面邏輯都不一樣，搜尋方式也不同。

1.2.4 現有的語音助理

市面上已經有 Siri、Google Assistant、Alexa 這些語音助理。理論上，可以對 Siri 說「提醒我明天買牛奶」，它就會設定提醒。

但實際使用後，我們發現這些工具在個人資訊管理上有很大的侷限：

記帳功能幾乎不存在

對 Siri 說「記錄今天午餐花了 150 元」，它會困惑地詢問要不要上網搜尋。即使透過「捷徑」功能勉強讓它呼叫第三方記帳 App，體驗也很差：需要事先

設定複雜的自動化流程，而且每次都要用固定的句型。

資料還是碎片化的

即使 Siri 能幫設定提醒，這些提醒還是在「提醒事項」App 裡；行事曆在「行事曆」App；備忘錄在「備忘錄」App。使用者還是要在不同 App 之間切換查看資訊。

更麻煩的是，想問「這個月我花了多少錢」時，Siri 完全幫不上忙，因為它根本不知道使用者的支出資料在哪裡。

無法學習個人習慣 Siri 不會記住使用者的分類習慣。每次說「午餐 150」，它都不知道該怎麼處理。

跨平台的困境

使用者用 iPhone，但工作電腦是 Windows。Siri 只能在 Apple 裝置上用，Google Assistant 主要在 Android 和 Google 生態系，兩邊的資料無法同步。

1.3 研究動機

如果有一個系統，專門為個人資訊管理設計，它應該要有：

1. 統一的介面：所有功能（記帳、待辦、行事曆、備忘錄）都在同一個對話視窗裡完成，不用切換 App。
2. 真正的記帳功能：不只是「可以記帳」，而是有完整的類別管理、統計分析、預算控制。
3. 跨平台：手機、電腦、平板都能用，資料即時同步。
4. 自然語言理解：不需要固定句型，可以說「今天中午跟同事吃飯大概 250 左右」這種模糊的表達。

現有的語音助理是「萬能型」的助理，什麼都會一點，但每個功能都不夠深入。我們想要的是一個「專業型」的助理，專精於個人資訊管理這件事。

現在，大型語言模型（LLM）的出現改變了遊戲規則：

- 更好的語言理解：LLM 可以理解各種表達方式，處理模糊的語句（「大概

250 左右」)。

- zero-shot 學習：不需要大量訓練資料，只要給 LLM 描述每個功能，它就能理解。
- 更容易開發：不需要訓練複雜的 NLP 模型，直接用 API 就能實現。

結合現代的通訊平台（如 Telegram)，我們可以快速開發一個跨平台、低門檻、專注於 PIM 的對話式系統。

這就是這個專案的目標：用 LLM 打造一個真正專注於個人資訊管理的對話式助理。

2. 相關文獻

2.1.1 LLM 驅動的意圖識別

Parikh et al. (2023) 在對話式 NLU 系統的研究中發現,僅透過在 prompt 中提供意圖描述,LLM 就能準確識別使用者意圖,無需額外的模型訓練或微調。該研究在意圖數量為 9 個的場景中,零樣本準確率高達 89.3%。

2.1.2 實際部署的最佳實踐

Arora et al. (2024) 針對任務導向對話系統的研究指出,LLM 的意圖識別性能受兩個因素影響:意圖數量(越少越好)和意圖範圍(細粒度的意圖定義比廣泛的定義表現更好)。研究建議採用細粒度的意圖定義,並保持較小的標籤空間。該研究顯示,Claude v3 Haiku 在真實場景測試中平均 F1 分數達到 0.736,延遲僅為 1.697 秒,證明了這種方法在實際應用中的可行性。

2.1.3 本專案的技術選擇

基於上述研究,本專案採用 LLM + 意圖描述的零樣本方法。本系統預計包含 10-15 個主要意圖,符合研究中表現最佳的範圍。意圖定義明確且不重疊,採用細粒度設計。根據 Parikh et al. 在 9 個意圖場景下達到約 89% 的準確率,

以及 Arora et al. 在實際部署中觀察到約 74% 的 F1 分數,預期本專案在類似應用情境下的表現應介於兩者之間。

3. 系統功能

我們預計實作四大核心功能模組，涵蓋個人資訊管理的主要需求。使用者可以透過自然語言或是指令與系統互動。

3.1 使用範例

- 記帳功能

```
使用者：「今天中午跟同事吃飯 250」
系統：✓ 已記帳
      金額：NT$ 250
      類別：飲食
      日期：2025-11-09
```

- 待辦事項功能

```
使用者：「下週五前要完成期末報告」
系統：✓ 新增待辦事項
      標題：完成期末報告
      截止：2025-11-15
      優先級：中
      提醒：會在到期前一天提醒
```

- 行事曆功能

```
使用者：「明天下午 3 點跟客戶開會」
系統：✓ 已新增行事曆事件
      事件：客戶會議
      時間：2025-11-10 15:00
      提醒：15 分鐘前
```

- 備忘錄系統

```
使用者：「記住我的車位是 B325」
系統：✓ 已記錄備忘錄
      內容：車位 B325
      標籤：#停車 #車位
```

3.2 核心功能模組

3.2.1 記帳系統

- 基礎收支功能
- 動態類別：我們將設計多個泛用類別，動態的類別判斷，根據描述自動分類，不需使用者預先定義類別，也可以在現有類別不適用時，自動為使用者生成類別。
- 支出統記與分析：按日、週、月統計，分析類別佔比，並可以選擇讓 LLM 分析結果，讓使用者更容易了解內容。

3.2.2 待辦事項管理

- 自動識別時間，理解「下週五」「明天」等相對時間
- 智能優先級判斷：根據截止時間和關鍵字自動判斷重要性。
- 彈性的提醒設定
- 逾期自動提醒
- 完成狀態追蹤

3.2.3 行事曆與事件

- 支援單次和重複事件
- 彈性的提醒設定
- 重複事件管理

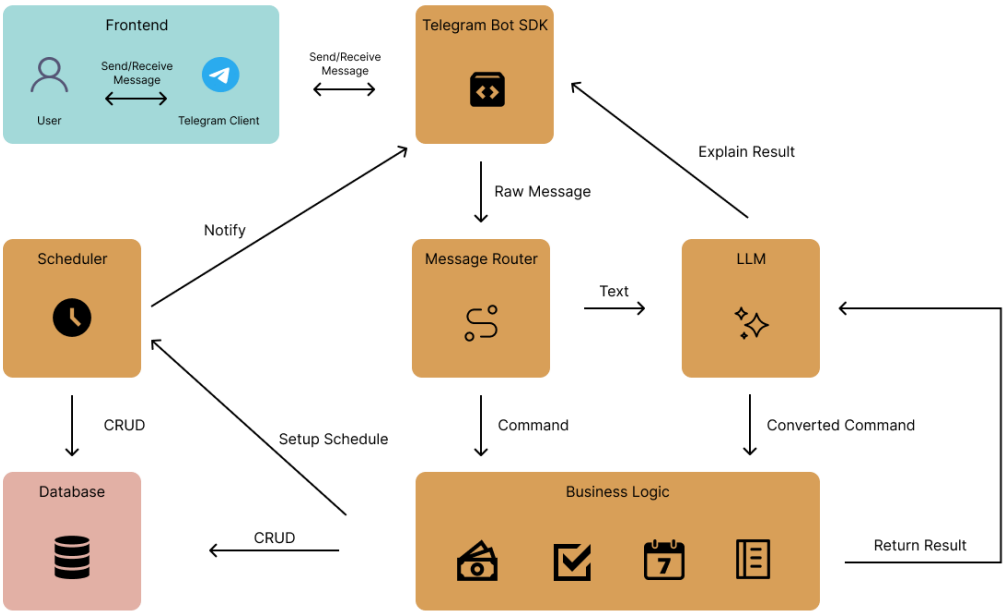
3.2.4 備忘錄系統

- 快速記錄任意資訊
- LLM 自動提取關鍵字與標籤
- 全文關鍵字搜尋
- 支援模糊查詢

3.3 Tech Stack

- 語言：Python 3.12+
- 資料庫：PostgreSQL
- LLM API：Anthropic Claude / Google Gemini / xAI Grok（待評估後選定）

3.4 系統架構圖



3.5 組件說明

3.5.1 Frontend（前端）

使用者透過 Telegram 客戶端進行互動，負責訊息的發送與接收。

3.5.2 Telegram Bot SDK

與 Telegram 平台的介面層，負責處理訊息收發協定。

3.5.3 Message Router（訊息路由器）

接收來自 Telegram Bot SDK 的原始訊息，判斷訊息並分流：

- Text：自然語言文字，送至 LLM 判斷語意
- Command：指令型訊息(/status, /list_todo 等)直接送至 Business Logic 層處理。

3.5.4 LLM（大型語言模型）

LLM 元件扮演的是「函數調度器」的角色負責自然語言理解，識別意圖、抽取參數等後，調用 Business Logic 層函數，並解釋結果返回給 Telegram Bot SDK。

3.5.5 Business Logic（業務邏輯層）

系統的核心處理層，包含四大功能模組：財務管理、任務管理、事件管理、知識管理，當被調用時執行業務邏輯，結果將返回 LLM 層。

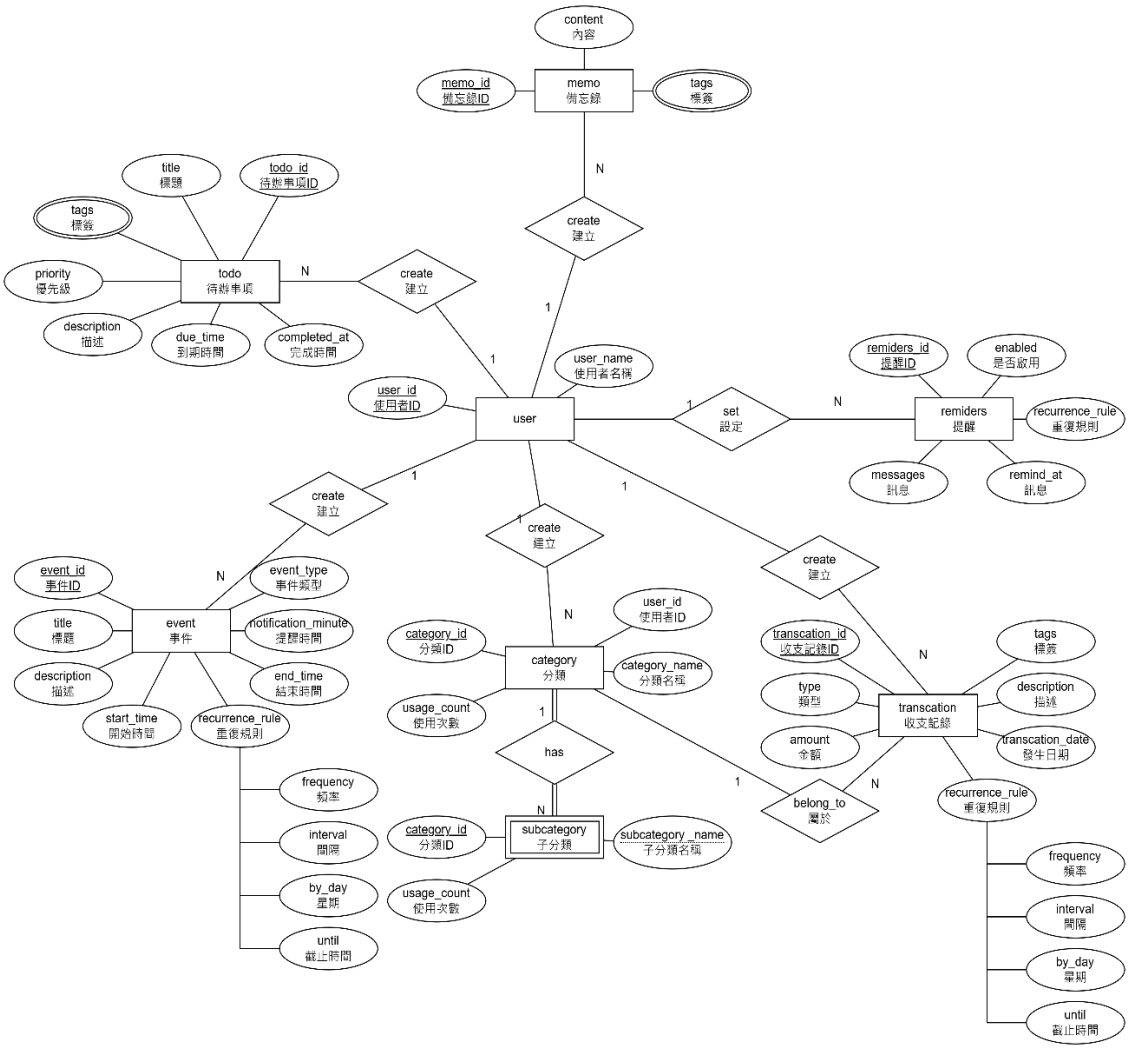
3.5.6 資料庫

使用 PostgreSQL 關聯式資料庫，接收來自業務邏輯層與排程器的增刪改查。

3.5.7 Scheduler（排程器）

負責需要主動發送訊息的任務，例如通知等。可以定時向資料庫查詢任務，確保一致性。

4. 資料庫 ERD 圖


content
內容
memo_id
備忘錄ID
memo
備忘錄
tags
標籤
N
create
建立
1
title
標題
todo_id
待辦事項ID
tags
標籤
priority
優先級
todo
待辦事項
N
create
建立
description
描述
due_time
到期時間
completed_at
完成時間
1
user_name
使用者名稱
user_id
使用者ID
user
1
set
設定
N
remiders_id
提醒ID
enabled
是否啟用
remiders
提醒
recurrence_rule
重復規則
messages
訊息
remind_at
訊息
1
create
建立
1
create
建立
1
create
建立
event_id
事件ID
N
event_type
事件類型
title
標題
event
事件
notification_minute
提醒時間
description
描述
end_time
結束時間
start_time
開始時間
recurrence_rule
重復規則
frequency
頻率
interval
間隔
by_day
星期
until
截止時間
N
user_id
使用者ID
category_id
分類ID
category
分類
category_name
分類名稱
usage_count
使用次數
1
has
N
category_id
分類ID
subcategory
子分類
subcategory_name
子分類名稱
usage_count
使用次數
transcation_id
收支記錄ID
N
tags
標籤
type
類型
transcation
收支記錄
description
描述
amount
金額
transcation_date
發生日期
1
belong_to
屬於
N
recurrence_rule
重復規則
frequency
頻率
interval
間隔
by_day
星期
until
截止時間

5. 後續進度規劃表與組員分工表

5.1 進度規畫

我們將開發進度分成 7 個里程碑，並預計一周能夠達成一項里程碑：

里程碑	名稱	主要任務	成果
1	基礎架構完成	開發環境設定 資料庫設計與建置 Telegram Bot 基本框架 LLM API 整合測試	可運作的 Bot 框架 + 完整資料庫
2	記帳功能完成	LLM 意圖識別與參數提取 交易記錄 CRUD 功能 類別判斷與學習機制 基本查詢功能	完整的智能記帳系統
3	待辦事項管理	待辦事項 CRUD 時間解析邏輯 優先級判斷 逾期提醒	完整的待辦事項管理系統
4	行事曆與事件	行事曆事件管理 重複事件邏輯 事件查詢功能	行事曆功能可用
5	提醒系統	提醒通知機制 背景任務排程 一次性與重複提醒 提醒管理功能	完整的提醒系統
6	備忘錄與進階功能	備忘錄系統 自動標籤提取 全文搜尋 預算管理 資料匯出	所有核心功能完成
7	測試與交付	功能測試與 Bug 修復 使用者測試 系統文件撰寫 部署與上線	可交付的完整系統

5.2 組員分工表

成員	主要職責
蘇泓叡	資料庫設計、核心 API 開發、系統架構
邱柏愷	LLM 整合、意圖識別、自然語言處理
李承育	Telegram Bot 介面、使用者互動、訊息處理
徐慶峰	輔助開發、進階功能、文件撰寫、專案管理

6. 參考文獻

Parikh, S., Tiwari, M., Tumbade, P., & Vohra, Q. (2023, July). Exploring zero and few-shot techniques for intent classification. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track) (pp. 744-751).

Arora, G., Jain, S., & Merugu, S. (2024). Intent detection in the age of llms. *arXiv preprint arXiv:2410.01627*.