國立臺北商業大學

資 訊 管 理 系

113資訊系統專案設計

**系統手冊**

****

**組 別：第113402組**

**題 目：租中自有黃金屋**

**指導老師：蒯思齊老師**

**組 長：11046008 郭育廷**

**組 員：11046016 廖芸珮 11046028 蘇櫟**

**11046030 黃薪橙 11046034 陳玉函**

**中華民國113年05月29日**

目錄

[第1章 前言 1](#_Toc175065148)

[1-1 背景介紹 1](#_Toc175065149)

[1-2 動機 1](#_Toc175065150)

[1-3 系統目的與目標 2](#_Toc175065151)

[1-4 預期成果 2](#_Toc175065152)

[第2章 營運計畫 3](#_Toc175065153)

[2-1 可行性分析 3](#_Toc175065154)

[2-2 商業模式－Business model 5](#_Toc175065155)

[2-3 市場分析－STP 5](#_Toc175065156)

[2-4 競爭力分析SWOT-TOWS 6](#_Toc175065157)

[第3章 系統規格 8](#_Toc175065158)

[3-1 系統架構 8](#_Toc175065159)

[3-2 系統軟、硬體需求與技術平台 9](#_Toc175065160)

[3-3 使用標準與工具 10](#_Toc175065161)

[第4章 專案時程與組織分工 14](#_Toc175065162)

[4-1 專案時程 14](#_Toc175065163)

[4-2 專案組織與分工 15](#_Toc175065164)

[第5章 需求模型 20](#_Toc175065165)

[5-1 使用者需求 20](#_Toc175065166)

[5-2 使用個案圖 20](#_Toc175065167)

[5-3 使用個案描述 21](#_Toc175065168)

[5-4 分析類別圖 24](#_Toc175065169)

[第6章 設計模型 25](#_Toc175065170)

[6-1 循序圖(Sequential diagram)或通訊圖(Communication diagram) 25](#_Toc175065171)

[6-2 設計類別圖 30](#_Toc175065172)

[第7章 實作模型 31](#_Toc175065173)

[7-1 佈署圖(Deployment diagram) 31](#_Toc175065174)

[7-2 套件圖(Package diagram) 31](#_Toc175065175)

[7-3 元件圖(Component diagram) 33](#_Toc175065176)

[7-4 狀態機(State machine) 33](#_Toc175065177)

[第8章 資料庫設計 37](#_Toc175065178)

[8-1 資料庫關聯圖 37](#_Toc175065179)

[8-2 表格及其Meta data 39](#_Toc175065180)

[第9章 程式 43](#_Toc175065181)

[9-1 元件清單及其規格描述 43](#_Toc175065182)

[9-2 其他附屬之各種元件 46](#_Toc175065183)

[第10章 47](#_Toc175065184)

[10-1 測試計畫 47](#_Toc175065185)

[10-2 測試個案與測試結果資料 47](#_Toc175065186)

[第11章 操作手冊 47](#_Toc175065187)

[第12章 使用手冊 47](#_Toc175065188)

[第13章 感想 47](#_Toc175065189)

[第14章 參考資料 47](#_Toc175065190)

[附錄一 會議記錄 50](#_Toc175065191)

[附錄二 問卷題目 54](#_Toc175065192)

[附錄三 評審問題回覆 56](#_Toc175065193)

**圖目錄**

[▲圖 2-1‑1 112年度台北大專院校人數租屋比例圖 3](#_Toc174309198)

[▲圖 2-1‑2問卷調查-租房人數概況圖 4](#_Toc174309199)

[▲圖 2-1‑3問卷調查-各平台缺點 4](#_Toc174309200)

[▲圖 2-1‑5 112年度租屋痛苦指數 4](#_Toc174309201)

[▲圖 3-1‑1系統架構圖 8](#_Toc174309202)

[▲圖 3-1‑2系統流程圖 9](#_Toc174309203)

[▲圖 3-2‑1 2023年台灣行動作業系統占比圖 9](#_Toc174309204)

[▲圖 3-2‑2歷年來全球作業系統占比圖 10](#_Toc174309205)

[▲圖 4-2‑1 Github頁面 19](#_Toc174309206)

[▲圖 5-2‑1使用個案圖 20](#_Toc174309207)

[▲圖 5-4‑1分析類別圖 24](#_Toc174309208)

[▲圖 6-1‑1循序圖－註冊會員 25](#_Toc174309209)

[▲圖 6-1‑2循序圖-登入 25](#_Toc174309210)

[▲圖 6-1‑3循序圖-更改密碼 26](#_Toc174309211)

[▲圖 6-1‑4循序圖-房屋查找 26](#_Toc174309212)

[▲圖 6-1‑5循序圖-訂閱 27](#_Toc174309213)

[▲圖 6-1‑6循序圖-修改個人資料 27](#_Toc174309214)

[▲圖 6-1‑7循序圖-收藏房屋 28](#_Toc174309215)

[▲圖 6-1‑8循序圖-編輯我的收藏 28](#_Toc174309216)

[▲圖 6-1‑9循序圖-瀏覽我的收藏 29](#_Toc174309217)

[▲圖 6-1‑10循序圖-查看瀏覽紀錄 29](#_Toc174309218)

[▲圖 6-2‑1設計類別圖 30](#_Toc174309219)

[▲圖 7-1‑1佈署圖 31](#_Toc174309220)

[▲圖 7-2‑1套件圖－爬蟲Get detail 31](#_Toc174309221)

[▲圖 7-2‑2套件圖－爬蟲Get image 31](#_Toc174309222)

[▲圖 7-2‑3套件圖－爬蟲Get URL 31](#_Toc174309223)

[▲圖 7-2‑4套件圖－串接前後端Flask API 32](#_Toc174309224)

[▲圖 7-2‑5套件圖－Line message API 31](#_Toc174309225)

[▲圖 7-2‑6套件圖－後端 重複資料判斷 32](#_Toc174309226)

[▲圖 7-2‑7套件圖－資料存入MySQL 32](#_Toc174309227)

[▲圖 7-2‑8套件圖－圖片存入MySQL 32](#_Toc174309228)

[▲圖 7-3‑1元件圖 33](#_Toc174309229)

[▲圖 7-4‑1狀態機－登入 33](#_Toc174309230)

[▲圖 7-4‑2狀態機－註冊 34](#_Toc174309231)

[▲圖 7-4‑3狀態機－忘記密碼 34](#_Toc174309232)

[▲圖 7-4‑4狀態機－修改密碼 35](#_Toc174309233)

[▲圖 7-4‑5狀態-編輯訂閱標籤 35](#_Toc174309234)

[▲圖 7-4‑6狀態機－搜尋房源 36](#_Toc174309235)

[▲圖 8-1‑1關聯式資料庫ER圖-房屋資料 37](#_Toc174309236)

[▲圖 8-1‑2關聯式資料庫ER圖-使用者資料 37](#_Toc174309237)

[▲圖 8-1‑3關聯式資料庫ER圖-未來規劃 38](#_Toc174309238)

**表目錄**

[表 1-2‑1各租屋平台之優點 1](#_Toc167872532)

[表 2-1‑1問卷統計概況 4](#_Toc167872535)

[表 2-2‑1 商業模式-Business model 5](#_Toc167872537)

[表 2-4‑1競爭力分析SWOT-TOWS 7](#_Toc167872538)

[表 3-2‑1系統硬體需求-手機 10](#_Toc167872543)

[表 3-3‑1使用標準與工具表 12](#_Toc167872544)

[表 4-1‑1專案時程表 13](#_Toc167872545)

[表 4-2‑1專業組織與分工表 14](#_Toc167872546)

[表 4-2‑2專題成果工作內容與貢獻度表 16](#_Toc167872547)

[表 5-1‑1使用者需求表 20](#_Toc167872559)

[表 5-3‑1房屋查找 21](#_Toc167872561)

[表 5-3‑2訂閱功能 21](#_Toc167872562)

[表 5-3‑3瀏覽房屋詳細資訊 21](#_Toc167872563)

[表 5-3‑4使用Line智能助手-規劃中 22](#_Toc167872564)

[表 5-3‑5收藏房源 22](#_Toc167872565)

[表 5-3‑6瀏覽歷史紀錄 22](#_Toc167872566)

[表 5-3‑7房源管理(新增) 23](#_Toc167872567)

[表 5-3‑8房源管理(編輯) 23](#_Toc167872568)

[表 5-3‑9推播訂閱資訊 24](#_Toc167872569)

[表 8-2‑1資料表 40](#_Toc167872603)

[表 8-2‑2資料表描述－01房屋詳細資料 40](#_Toc167872604)

[表 8-2‑3資料表描述－02房屋設備 41](#_Toc167872605)

[表 8-2‑4資料表描述－03房屋地址 41](#_Toc167872606)

[表 8-2‑5資料表描述－04房屋仲介 42](#_Toc167872607)

[表 8-2‑6資料表描述－05會員資料 42](#_Toc167872608)

[表 8-2‑7資料表描述－06使用者頭貼 43](#_Toc167872609)

# 前言

## 背景介紹

在這個快速變遷的社會中，租屋問題一直是許多學子關心的議題之一。隨著學生流動比率的增加，學生們尋找優質且合適的租屋處變得越來越具有挑戰性。

因此，我們先以大專院校學生以及年輕族群租屋的角度出發，旨在利用網路上現有的租屋平台及各種來源，改善我們發現的問題為有租屋需求的人提供更便捷、更優質的租屋資訊。我們將探討現有租屋平台的優缺點，分析使用者租屋需求的特點，並提出相應的解決方案，讓使用者能夠更輕鬆地找到符合自身需求的理想租屋處，為租屋族的生活品質帶來更多可能性。

## 動機

傳統的租屋方式往往是透過房屋仲介或是口耳相傳，但這些方式存在著資訊不對稱、仲介費用高昂等問題，使得租屋過程充滿了不確定性和風險。本組所有成員皆是離開家鄉到台北求學的學生，所以我們深知找房子的迫切需求和困難之處。每到租房季，總是要提前好幾個月尋找合適的租屋處，需要花費大量的時間、心力以及金錢。

目前市面上存在著許多租屋平台，每個平台都有自己的特點和優勢，但也同時存在著一些缺點和限制。例如：有些平台資訊不夠詳細，無法全面了解房屋情況；有些平台介面複雜，使用起來不夠直觀；還有些平台不會篩選刊登者，讓我們難以信任其中提供的資訊。

如何在眾多的選擇中找到最適合自己的租屋處儼然成為了新的挑戰，所以我們希望通過整合多個租屋平台的資訊和資源，結合各平台優點以及改善缺點來提供一個更全面、更可靠的租屋平台，讓北漂學子抑或是有租屋需求的人能夠更輕鬆地找到符合需求的房子。透過整合，我們也可以更有效地管理和比較不同房源的價格、設備、地理位置等資訊，幫助使用者做出更明智的租屋決策。

▼表 1-2‑1各租屋平台之優點

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 項目 平台 | 黃金屋 | 591房屋交易 | 好房網 | 樂屋網 | 租租通 | FB社團 |
| 有APP | V | V | V | V |  | V |
| 篩選功能完善 | V | V | V |  | V |  |
| 訂閱推送功能 | V | V | V(買房) | V |  |  |
| 過濾重複刊登 | V |  |  |  |  |  |
| Line智能助手 | V |  |  |  |  |  |

## 系統目的與目標

本次專題目的在於提供一個更完善的租屋平台，從使用者的角度出發，滿足租客對於租屋需求的多樣性和個性化。透過提供更全面的篩選條件，簡化租屋搜尋的過程，有效過濾重複刊登物件，讓使用者能夠更快速、更有效地找到符合其條件和偏好的租屋選項。以滿足不同租客的需求。同時，我們將設計更直覺、易用的使用者界面，減少使用者在租屋過程中的誤解和困惑，從而提高整體的使用者體驗。

## 預期成果

我們預期系統可以過濾重複刊登物件、幫使用者找出更精確且多面向的房源資訊等功能，從而提高租客對於房源資訊的真實性和可信度。減少篩選器的限制，使篩選功能更完善，貼近租客需求。同時透過AI技術，結合Line聊天機器人與ChatGPT API，提供租客一個智能化的客服，通過線上對話方式幫助租客解決問題、提供建議，從而達到「AI帶路，租屋不迷路」。

# 營運計畫

## 可行性分析

1. 時程可行性：因學校專題課程時間限制，故本組無法將系統功能開發到非常完善，於是先篩選出鄰近本校的幾個地區(中正區、大安區、萬華區、士林區、中山區)作為本次專題的目標資料來源。爬蟲的部分先針對有著最多房源的平台-591租屋網作為初始資料以支持系統展示。
2. 技術可行性：目前環境存在的技術已足夠支援本系統之開發，例如：使用爬蟲將房屋的相關資訊以及會員資料存到MySQL資料庫、Neo4j圖形資料庫來處理房源之間的複雜關聯，優化數據存儲與查詢、LINE Message API 開放使用者回答問題並且調用資料庫資料向使用者提供房屋推薦、使用Ckiptagger與Bert進行分詞與相似度分析、利用YOLOv8和CLIP進行圖像處理和物體檢測，以提升系統對圖像數據的處理能力、Microsoft Azure建構雲端服務提供虛擬機、Flutter框架以及Dart程式語言撰寫使用者介面、利用Flask建立API連接前端App及後端MySQL資料庫。
3. 市場可行性：傳統的租屋方式如房屋仲介和口耳相傳存在諸多不便，像是信息不對稱和高額的仲介費等等，這使得租屋過程變得不確定且風險較高。而這些問題在台北尤為突出，特別是對於從其他地方來到台北生活的年輕族群。所以我們特別根據在台北的大專院校租屋人數比例來評估市場可行性。



▲圖 2-1‑1 112年度台北大專院校人數租屋比例圖

我們也發放問卷請身邊的朋友填寫，針對是否有租屋來進行調查，發現大部分都是在外租屋；也特別針對現今租屋平台之缺點調查，分析出大家覺得難用的缺點，而這些缺點大部分都是本組研發此系統預期要改善的重點。

|  |  |
| --- | --- |
| ▲圖 2-1‑2問卷調查-租房人數概況圖 | ▲圖 2-1‑3問卷調查-各平台缺點 |

▼表 2-1‑1問卷統計概況

|  |  |  |  |
| --- | --- | --- | --- |
| 問卷針對年齡範圍：大部分是18-22歲；少部分為23-35歲 | | | |
| 預計共發放100份 | 實際回收78份 | 有效77份 | 無效1份 |
| \*並非只針對國立臺北商業大學之學生發放問卷 | | | |
| \*問卷相關題目列於附錄二 | | | |

1. 收益可行性：由於租屋市場需求的逐漸增長，尤其是對於台北來說，人口流動性高，加上房價昂貴，讓大多數人選擇租房居住。由下圖可知，台北租房的痛苦指數為全世界排名第19名，由此可知在臺北地區租到好房有一定的困難，所以我們提供一個好的租房系統讓使用者能夠得到更精確的篩選，以符合使用者需求，增加使用者使用本系統的意願，甚至是推薦給身邊的朋友，增加使用本系統的人口基數，為我們帶來可觀的收益。



▲圖 2-1‑4 112年度租屋痛苦指數

## 商業模式－Business model

▼表 2-2‑1 商業模式-Business model

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **關鍵合作夥伴**   * 房東 * 房仲 | **關鍵活動**   * 平台維護 * 房源資訊更新 * 出租人資訊 | **價值主張**   * 租屋知識 * 直觀簡潔的使用者介面 * LINE智能助手 * 節省尋找房屋的時間和精力 * 運用AI技術過濾及篩選房源 | | **顧客關係**   * 會員制 * 訂閱推送 * 客服服務 | **目標客戶**   * 有租屋需求之房客 * 資訊接受度較高之族群 |
| **關鍵資源**   * 租屋房源管理 * 刊登、出租系統 | **通路**   * 線上APP |
| **成本結構**   * 平台開發及維護費用 * 行銷費用 * 客服運營成本 | | | **收益流**   * 刊登收入 * 廣告收入 | | |

根據表2-2-1 商業模式-Business model，本組的目標為對於資訊接受度較高的租屋族群，以訂閱推送、Line智能助手為主要服務，吸引更多想租屋或者是租不到好房的租客來使用，提供更直觀簡潔的使用者介面，以節省尋找房屋的時間和精力，而本系統則以房東刊登費用及廣告收入為主要來源收入。

## 市場分析－STP

Segmentation市場區隔

* 依照**尋找房源習慣**區分:習慣使用租屋App的族群
* 依照**使用習慣**區分:對於資訊接受度較高之族群
* 依照**租屋經驗**區分:有/無

Targeting 適合的目標市場

* 對於資訊接受度較高之族群：本組將目標市場定為對於新技術、新知識和新資訊特別開放和敏感的人群，通常生活在數字化、網路化的環境中，對於新事物的接受度很高。
* 有出租需求之出租人：提供給房東一個平台，使他們的房源能夠更快出租。
* 喜歡簡潔的使用者介面：我們發現許多平台都存在著使用者介面不直觀的情形，所以設計更直觀簡潔的畫面，讓使用者節省時間及精力，能夠更快找到喜歡的物件。

Positioning 定位

* App內建整合好的租屋知識提供給使用者，不需要到網路找大量資料。
* 結合Line智能助手，提供一個線上的智能客服服務。
* 透過AI技術過濾及篩選重複房源

## 競爭力分析SWOT-TOWS

優勢(Strengths)：

* 新增Line智能助手，改善用戶支持與互動。
* 提供訂閱推送功能。
* 較直觀的使用者介面。
* 透過AI提供更精確的篩選器。
* 有效過濾重複刊登房源。

劣勢(Weakness)：

* 資金有限，線上智能客服無法提供24小時。
* 目前初入市場，使用者基數小。

機會(Opportunity)：

* 隨著租屋需求的增加，特別是位於都市區學生和年輕工作族群中，有機會擴大用戶基礎。
* 目前大眾對於資訊的接受度普遍提升，未來用戶的數量也會跟著成長

威脅(Threats)：

* 市場上已存在多個成熟的租屋平台，如591和好房網，需要在這些競爭對手中脫穎而出。
* 無法持續增加房源資料與擴展用戶基礎可能會降低競爭力。

▼表 2-4‑1競爭力分析SWOT-TOWS

|  |  |  |
| --- | --- | --- |
| 內部  外部 | 優勢(Strengths) | 劣勢(Weakness) |
| 機會(Opportunity) | 1. 提供更準確的房源推送，結合市場上租屋需求的增加，快速擴展客戶基礎。 2. 加強用戶互動提升服務滿意度，如：Line智能助手，特別針對資訊接受度較高之族群等重點客戶。 | 1. 儘管目前涵蓋房源有限，還是要逐步擴大範圍至其他區域，滿足更多使用者的租房需求。 2. 提升智能客服系統，使其能夠24小時提供，以應對不同時段的用戶需求。 |
| 威脅(Threats) | 1. 透過Line 智能助手及過濾重複房源等功能，提供其他平台無法比擬的特色客服，應對競爭對手威脅。 2. 透過AI提供更精確的篩選器，鎖定特定區域和族群，避免直接與大平台正面競爭。 | 1. 結合各種線上與線下管道，讓潛在客戶更快了解平台特色，減少競爭對手的影響。 2. 透過有效的過濾和分析，確保房源訊息真實性，提高用戶對平台的信任感。 |

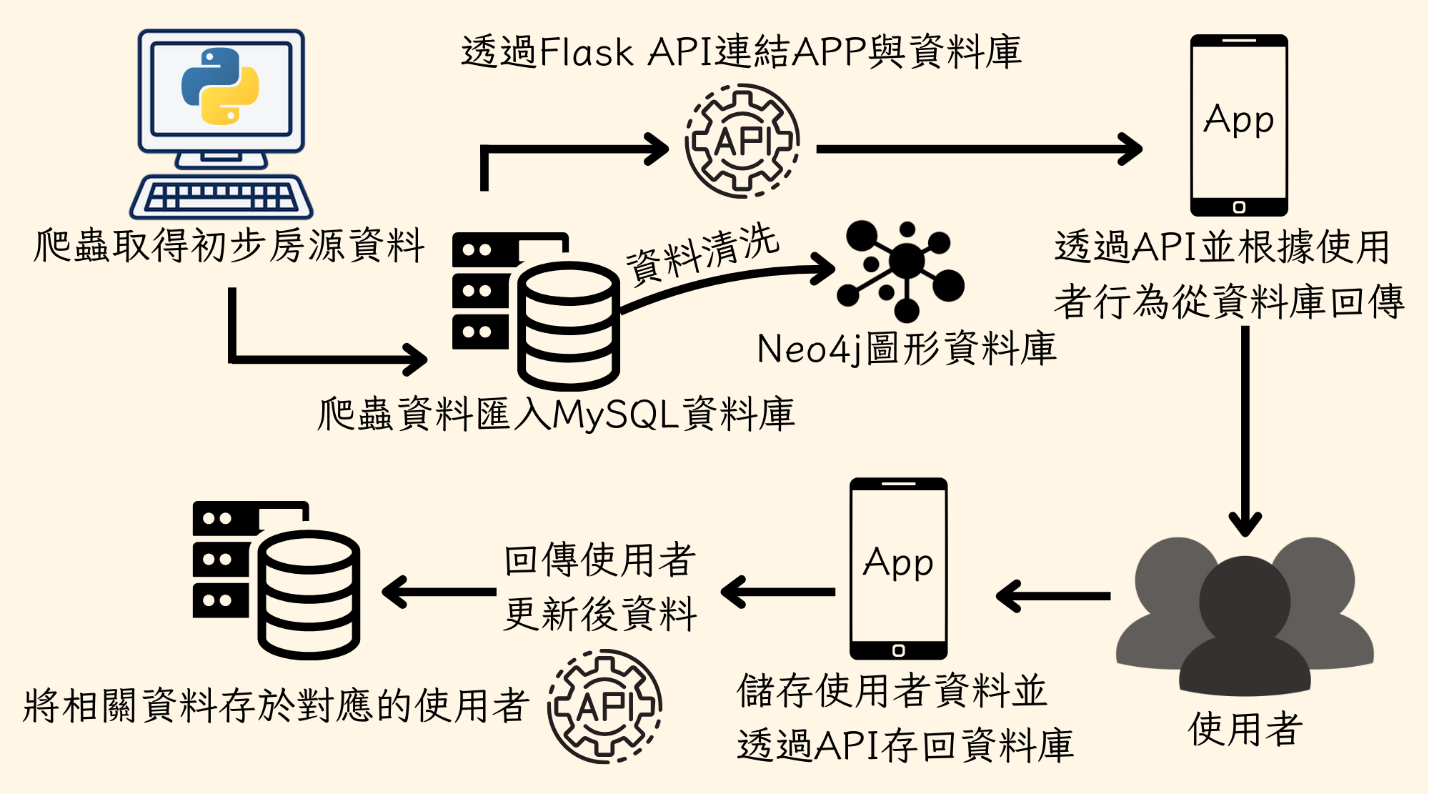
# 系統規格

## 系統架構

大部分的租屋平台都有著不同的優缺點，而我們的系統旨在結合多個平台的優點加以強化，以提供一個全面而可靠的租屋服務。先透過爬蟲技術從租屋平台獲取資訊作為初期運作的資料，並使用數據分析技術來評估房源。系統架構主要分為三個部分：

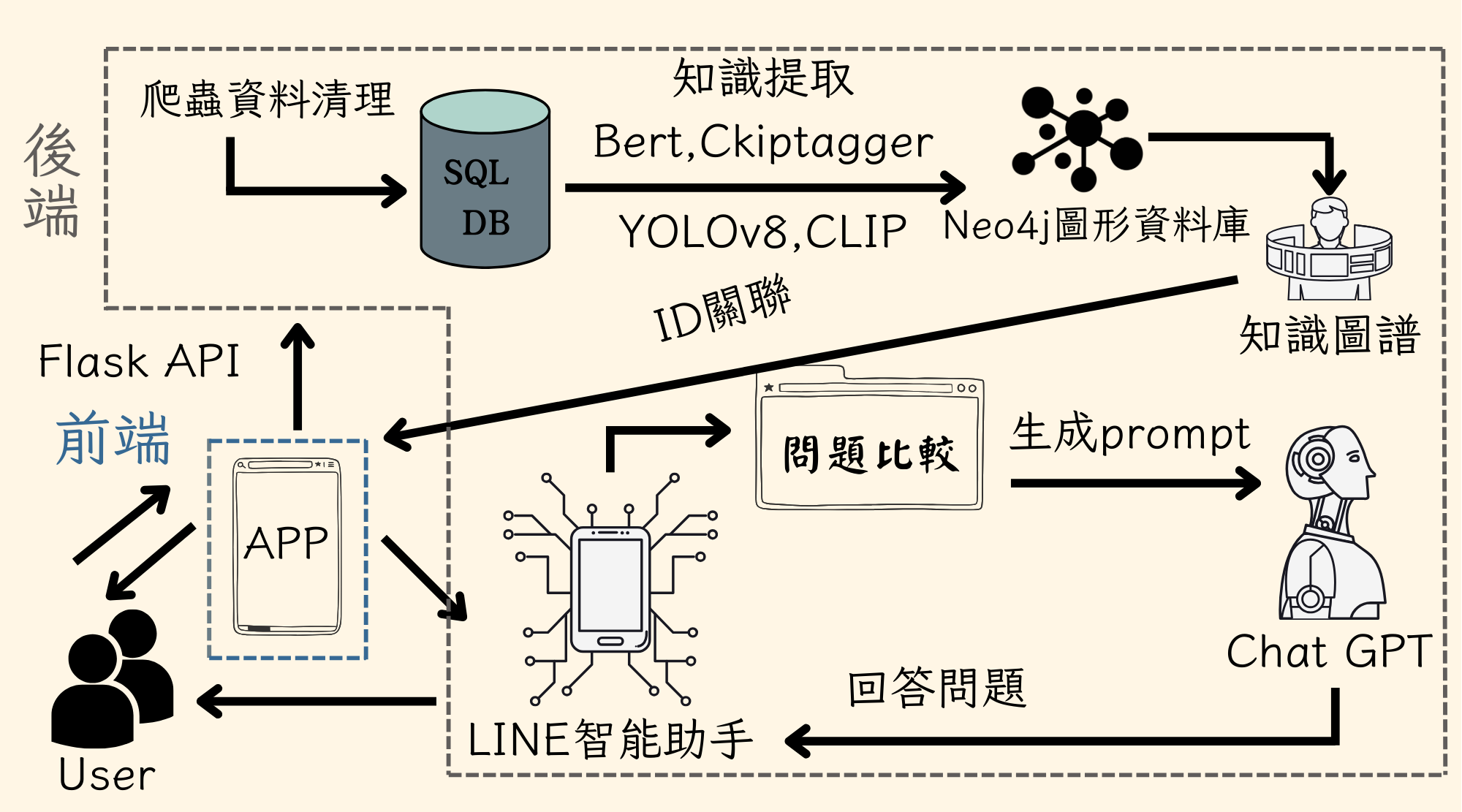
1. 前端介面：使用Flutter框架和Dart程式開發，提供用戶簡潔直觀的操作界面，讓用戶可以方便地瀏覽房源並進行篩選，節省時間以及精力。
2. 後端服務：利用Python爬取資料作為系統初期的資料來源、運用MySQL資料庫作為數據庫管理房源資料、儲存更新過後的使用者資料；Neo4j圖形資料庫來處理房源之間的複雜關聯，優化數據存儲與查詢。
3. 串接前端與後端：使用Flask建立API串接前端及後端MySQL資料庫。

以下為本組系統架構示意圖：



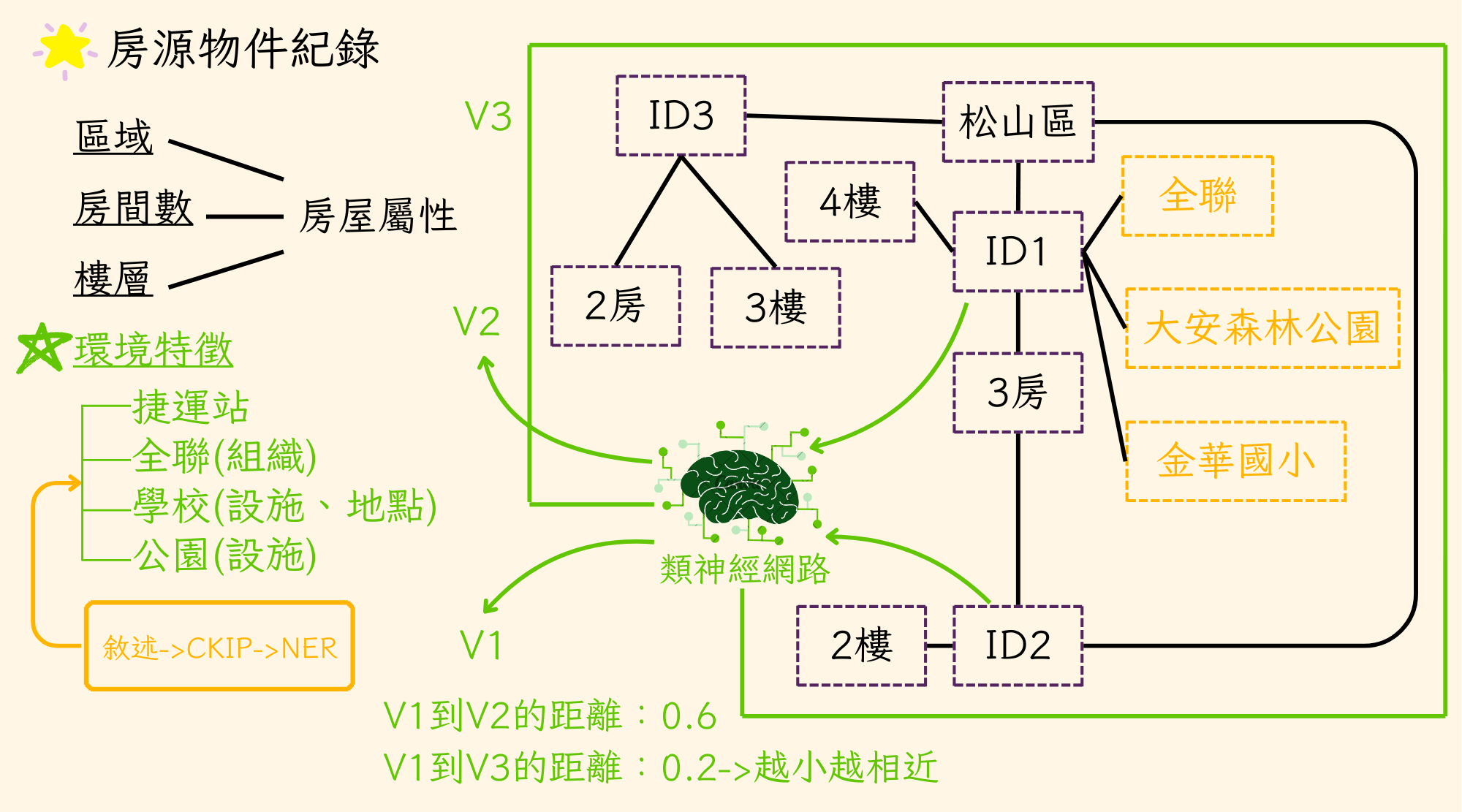
▲圖 3-1‑1系統架構圖

起初先運用爬蟲獲取資料作為此APP的租屋資訊來源，並將其存入資料庫。此外，系統利用自然語言處理(NLP)技術分析和比較房屋資訊的相似度，確保資料的準確性與相關性，有效過濾重複刊登房源。使用者登入APP後，可以收藏和查看自己曾瀏覽過的房源，也可以透過Line智能助手解決疑問。更進一步地，透過訂閱功能，結合AI技術將合適的房源推送給租客，提高房源的曝光率並加速租屋過程。這不僅為租客提供了個性化的房源推薦，也為房東帶來了更高的租屋效率。以下為本組系統流程示意圖：



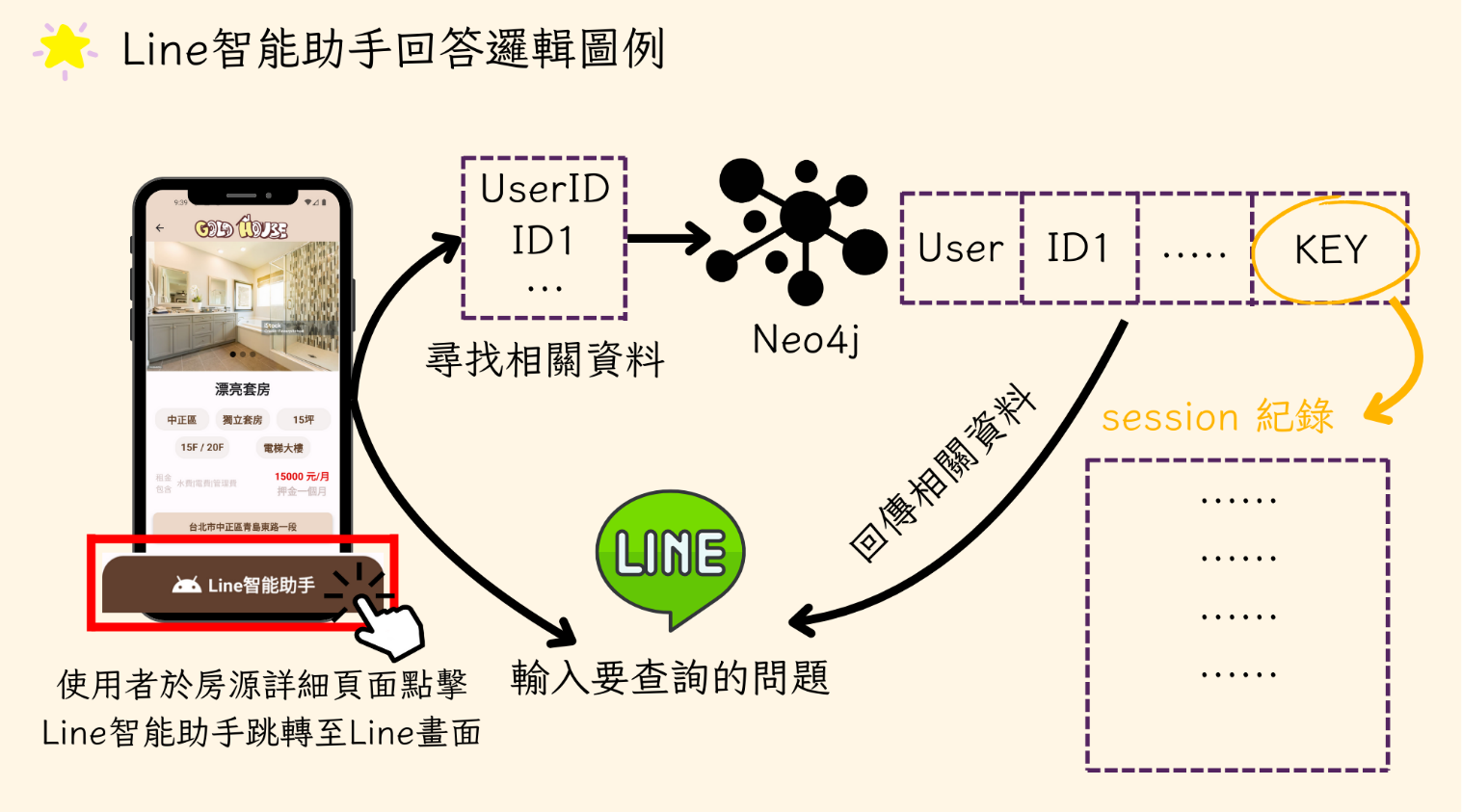
▲圖 3-1‑2系統流程圖

我們利用Neo4j來記錄房源物件和其環境特徵之間的關聯，包括房源物件的區域、房間數、樓層等屬性，以及周邊的環境特徵如捷運站、組織、設施、地點等。透過類神經網路和 CKIP 進行命名實體識別（NER），可以分析物件並標註環境特徵。V1、V2、V3 表示不同物件之間的相似度，數值越小代表相似度越高。這種視覺化有助於我們理解和分析房源及其周邊環境的相關性並將其整理為知識圖譜運用至Line智能助手中。



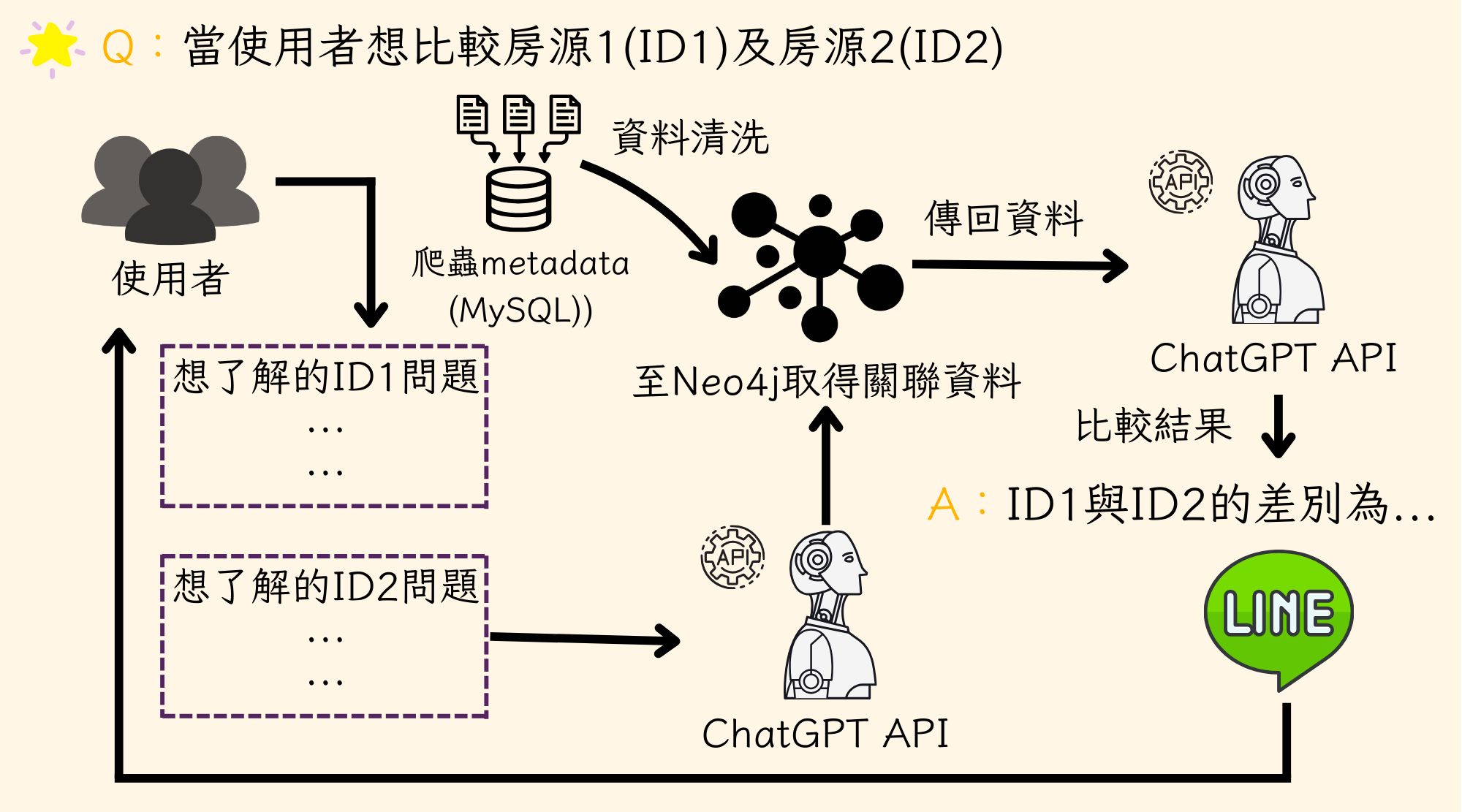
▲圖 3-1‑3 Neo4j圖譜解說圖例

當使用者在App內的房源詳細頁面點擊 "Line 智能助手" 按鈕後，系統會跳轉到 Line 畫面，使用者可以在該畫面輸入想要查詢的問題。系統根據使用者的 UserID 和所選房源(ID1)等信息，從Neo4j圖形式資料庫中搜尋相關資料，並將查詢結果返回給 Line智能助手。這些互動過程中產生的 session記錄會被保存下來，用於後續的查詢和回應。我們利用了 Neo4j 的資料關聯能力，提供使用者便捷的房源信息查詢服務，而不是像傳統的租屋平台，只能等待屋主的回應。



▲圖 3-1‑4 Line智能助手回答邏輯圖例

當使用者想比較房源1（ID1）及房源2（ID2）時，首先會提出想了解的問題。我們透過爬蟲技術取得Metadata並存入 MySQL資料庫做為系統初始運作資料來源，接著進行資料清洗。清洗後的資料被傳送至Neo4j，取得相關聯的資料。系統將這些資料傳送給 ChatGPT API 進行分析和比較，最後生成比較結果，並透過 Line將結果回傳給使用者，解答使用者對於 ID1 和 ID2 的差異問題。



▲圖 3-1‑5 Line智能助手解說圖例

## 系統軟、硬體需求與技術平台

根據StatCounter網站流量分析工具統計，在圖3-2-1 2023年台灣行動作業系統占比圖中iOS系統的比例高於Android系統，雖然我們選用的前端架構Flutter有跨平台的特性，能夠寫iOS以及Android系統，但礙於開發系統的限制，我們了解到iOS系統只能於macOS 作業系統環境下開發，本組成員皆是使用Windows作業系統，因此我們選擇開發Android系統。



▲圖 3-2‑1 2023年台灣行動作業系統占比圖

但在圖3-2-2歷年來全球作業系統占比圖的統計數據來說，起初，iOS系統的市場佔有率高於Android系統，但隨著時間的推移，Android系統的增長逐年上升，並在2012年首次超越iOS系統。此後，Android系統持續擴大其市場份額，穩定在一個較高的水平。這一趨勢顯示了Android系統的廣泛接受度和使用率，其開放性和多樣化的硬體選擇使其能夠吸引全球更多的用戶，也揭示了消費者偏好和技術發展如何影響智慧手機操作系統的流行度，故本組選擇Android系統作為主要開發系統。



▲圖 3-2‑2歷年來全球作業系統占比圖

考慮到市場上Android系統的普及和版本更新的趨勢，我們決定將手機的開發版本定為Android 8.0以上。此外，為了滿足現代通訊需求和提高用戶連接性，我們的設備將支持具備Wi-Fi、4G以及5G網路的用戶。這樣的配置將使我們的系統能夠適應快速變化的科技環境，同時滿足廣大用戶對高速網絡連接的需求。

▼表 3-2‑1系統硬體需求-手機

|  |  |
| --- | --- |
| 系統硬體需求-手機 | |
| 手機版本 | Android 8.0以上 |
| 網路需求 | Wi-Fi/4G/5G |

## 使用標準與工具

以下列出了本組所選用的開發工具與技術，並說明了每項的特點與選擇理由：

* Python：擁有簡潔且易於閱讀的語法，配備豐富的機器學習套件，特別適合進行高效能的爬蟲開發。我們選擇Python來構建爬蟲，以獲取並處理房源資料。
* MySQL：MySQL是一種免費的資料庫管理系統，用來儲存大量數據，我們使用此系統來儲存會員的相關資料以及房源資料。
* Neo4j：相較於傳統MySQL資料庫需要耗費大量時間、處理分析數據之間的關係。Neo4j 開發的原生圖形資料庫 (Native Graph Database)實現了使用知識圖譜的方式，將資料互相作關聯有助於比較各項資訊，協助我們達成AI辨識的功能。
* Flask：用Python編寫的輕量級Web應用框架，不僅容易上手且擴展性強，供更多自由度給開發者，我們使用Flask API來連接前端Flutter及後端MySQL資料庫。
* Flutter：為一個跨平台框架，利用Dart程式語言撰寫，能同時讓iOS與Android系統使用，也支援Web以及應用程式，為本組製作Android APP的工具。
* Android Studio：提供免費的整合式開發環境(IDE)，專門為Android平台的應用開發而設計。我們使用內建提供的模擬器，讓我們能在多種Android設備和配置上測試應用程式，而不需要實際擁有這些設備。
* Microsoft Azure：作為微軟的公有雲端服務平台，Azure支持多種雲端運算服務，包括基礎設施即服務(IaaS)、平台即服務(PaaS)甚至是軟體及服務(SaaS)。它允許我們在雲端輕鬆建置虛擬機和資料庫，並且相對於最多人使用的AWS具有更低的建置成本。因此，我們使用Microsoft Azure來進行資料庫以及虛擬機的架設。
* Diagrams.net：是一款流行的免費圖表繪製工具，廣泛用於創建各種UML圖表和視覺化，可以容易地與其他工具如：Google Drive、Dropbox、GitHub和OneDrive等整合，方便儲存和共享文件，本組使用此工具繪製個案圖、類別圖、循序圖、佈署圖、套件圖、元件圖、狀態機等。
* Anaconda：只要安裝後就不用煩惱各種 Python 安裝以及編輯器的問題，豐富的資料套件，能夠輔助各種資料邏輯運算，是一套很適合開發者的 Python 編輯軟體。我們使用Anaconda來解決版本問題。
* CKIPtagger：由台灣中央研究院語言學研究所開發的中文自然語言處理工具。這個工具包提供了詞性標註、命名實體識別和依存句法分析等功能。適合處理繁體中文文本。我們使用 CKIPtagger 進行中文分詞。
* BERT：由 Google AI Language 開發，是一種基於 Transformer 架構的預訓練語言模型。BERT 通過雙向編碼器表徵學習，可以更好地理解上下文信息，使其在各種自然語言處理任務中表現出色。廣泛應用於文本分類、問答系統、命名實體識別等任務。我們使用 BERT 進行分詞轉詞向量，再進行比對。
* YOLOv8：由 Ultralytics 開發的最新版本YOLO物體檢測模型。提供了即時物體檢測能力，能夠在單次前向傳播中同時進行多個物體的檢測和分類。適合用於各種實時物體檢測和圖像處理任務。我們使用 YOLOv8來偵測物品。
* CLIP：由 OpenAI 開發，是一種結合圖像和文本的預訓練模型。通過同時訓練圖像和文本，將它們映射到相同的向量空間，使圖像和文本之間的相似度可以被計算和比較。用於圖像搜索、圖像標註、跨模態檢索等應用。我們使用 CLIP 將物品轉換為文字描述。
* GitHub：作為軟體開發的合作平台，GitHub提供了團隊協作功能。透過Fork，我們能夠有效管理團隊成員的進度和動態，同時也便於檔案的審查和記錄，大大提升團隊的工作效率，為本組的專案進度管理工具。

▼表 3-3‑1使用標準與工具表

|  |  |
| --- | --- |
| 系統開發環境 | |
| 作業系統 | Windows11 |
| 程式開發工具 | |
| UI/UX設計 | Figma |
| 前端 | Flutter、 Android Studio |
| 後端 | Python、Microsoft Azure、My SQL、Neo4j、Flask |
| AI技術運用 | |
| 房屋文字相似度 | CKIPtagger、BERT |
| 房屋圖片相似度 | YOLOv8、CLIP |
| 開發平台 | Anaconda |
| 文件美工工具 | |
| 文件 | Microsoft Word、 Microsoft Excel |
| 簡報 | Microsoft PowerPoint、Canva |
| 圖檔 | Diagrams.net、Adobe Illustrator、Procreate、Flaticon |
| 影片 | Microsoft Clipchamp |
| 專案管理平台 | |
| 專案管理 | GitHub、Fork |
| 檔案存放 | GitHub、 Google Drive |

# 專案時程與組織分工

## 專案時程

根據專案工作流程，我們把專案分成14個項目，將專案彙整成甘特圖如下：

▼表 4-1‑1專案時程表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 113年 | | | | | | | | | | |
| 月份  項目 | 1月 | 2月 | 3月 | 4月 | 5月 | 6月 | 7月 | 8月 | 9月 | 10月 |
| Machine Learning 探討與實作 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 訂定主題 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 爬蟲探討與實作 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 文字資料預處理 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 文字相似度對比 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| UI/UX設計 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 頁面框架編寫 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 前端開發 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 前後端串接 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 資料庫建置 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 伺服器架設 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 文件製作 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 簡報製作 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
| 影片製作 |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

▅ 預期進度 / ▅ 實際進度

自今年1月起，我們開始鑽研與Machine Learning相關之技術，在每週的會議中會有成員對其負責章節的詳細報告。到了3月，我們轉向學習Python爬蟲技術，同時在學習Machine Learning的過程中，我們也邊尋找有興趣的主題來發想題目，用於後續的系統開發，確定主題後，我們也開始進行App的外觀設計與初步開發工作。

## 專案組織與分工

▼表 4-2‑1專業組織與分工表

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 項目 組員 | | 11046008  郭育廷 | 11036016  廖芸珮 | 11046028  黃薪橙 | 11046030  蘇櫟 | 11046034  陳玉函 |
| 後端開發 | 爬蟲-591(URL) |  |  | ● |  |  |
| 爬蟲-591(圖片) |  |  | ● |  |  |
| 爬蟲-591(詳細資料) |  |  | ● |  |  |
| 資料清洗 |  |  | ● |  |  |
| 匯入格式整理 |  |  | ● |  |  |
| 資料庫建置- MySQL | ● |  |  |  |  |
| MySQL-Data view | ● |  |  |  |  |
| MySQL-房屋資料 | ● |  |  |  |  |
| MySQL-會員資料 | ● |  |  |  |  |
| MySQL-刊登者資料 | ● |  |  |  |  |
| 資料分析-Ckiptagger |  |  |  | ● |  |
| Ckiptagger-WS 分詞 |  |  |  | ● |  |
| Ckiptagger-NER 實體辨識 |  |  |  | ● |  |
| Ckiptagger-POS 詞性標註 |  |  |  | ● |  |
| 資料分析- Bert |  |  |  | ● |  |
| Bert-address 地址 |  |  |  | ● |  |
| Bert-patten 房型 |  |  |  | ● |  |
| Pytorch transformers 斷詞轉換 |  |  |  | ● |  |
| 雲端(Azure)資料庫架設 | ● |  |  |  |  |
| 雲端(Azure)開發環境建置 | ● |  |  |  |  |
| Line API |  |  | ● |  |  |
| Chat GPT API |  |  | ● |  |  |
| Flask API |  |  | ● |  | ○ |
| 前端開發 | UI/UX設計 |  | ● |  |  |  |
| App介面設計 |  | ● |  |  |  |
| Flutter頁面框架編寫 |  |  |  |  | ● |
| App-登入與註冊 |  |  |  |  | ● |
| App-首頁 |  |  |  |  | ● |
| App-搜尋 |  |  |  |  | ● |
| App-刊登 |  |  |  |  | ● |
| App-訂閱 |  |  |  |  | ● |
| App-個人資料 |  |  |  |  | ● |
| App-收藏與瀏覽紀錄 |  |  |  |  | ● |
| 美術設計 | 色彩設計 |  | ● |  |  |  |
| Logo設計-  Adobe Illustrator | ● |  |  |  |  |
| Logo設計-Procreate |  |  |  | ● |  |
| 文件撰寫 | 統整 |  | ● |  |  |  |
| 第1章 前言 |  | ● |  |  |  |
| 第2章 營運計畫 |  | ● |  |  |  |
| 第3章 系統規格 |  | ● |  |  |  |
| 第4章 專題時程與組織分工 |  | ● |  |  |  |
| 第5章 需求模型 |  | ● |  | ○ |  |
| 第6章 設計模型 | ○ |  |  |  | ● |
| 第7章 實作模型 |  |  | ● | ○ |  |
| 第8章 資料庫設計 | ● |  | ○ |  |  |
| 第9章 程式 |  |  |  |  |  |
| 第10章 測試模型 |  | ● |  |  | ○ |
| 第11章 操作手冊 |  | ● |  |  | ○ |
| 第12章 使用手冊 |  | ● |  |  | ○ |
| 競賽相關文件與準備 |  | ● |  |  |  |
| 報告 | 簡報製作 |  | ● |  | ○ |  |
| 影片製作 |  | ○ | ● | ○ |  |

● 主要負責人 / ○ 次要負責人

▼表 4-2‑2專題成果工作內容與貢獻度表

|  |  |  |  |
| --- | --- | --- | --- |
| 序號 | 姓名 | 工作內容 | 貢獻度 |
| 1 | 組長  郭育廷 | 雲端伺服器：Azure虛擬機架設、虛擬資料庫建置  資料庫：MySQL及Neo4j資料庫建置、設計及管理  美編：logo 設計、名牌製作  文件：ER-model、循序圖繪製 | 20 % |
| 2 | 組員  廖芸珮 | App：UI/UX設計、介面設計、色彩設計  美編：簡報製作及統整  影片：影片構思、美化  文件：CH1-CH5及統整、競賽相關文件與準備 | 18 % |
| 3 | 組員  黃薪橙 | API：設計及管理、串聯資料庫與APP資料  LINE bot：LINE 聊天機器人架設  爬蟲：爬取租屋網站房屋資訊（591租屋網）  影片：影片編劇、拍攝  文件：部署圖、套件圖  NLP：使用CLIP和YOLOv8來偵測圖片物品並轉為文字敘述 | 22 % |
| 4 | 組員  蘇櫟 | NLP：使用Ckiptagger和Bert分析資料、使用CLIP和YOLOv8來偵測圖片物品並轉為文字敘述  美編：logo繪製  影片：影片剪輯、後製  文件：使用個案圖繪製、使用個案描述、分析類別圖繪製、元件圖繪製、狀態機繪製 | 19 % |
| 5 | 組員  陳玉函 | APP : 頁面框架編寫、Android 應用程式開發  API ：將應用程式介面框架與API整合  文件撰寫：循序圖繪製 | 21 % |
|  | | | 總計：100% |

在團隊討論完共同感興趣的主題後，我們將專題劃分為五大核心部分：後端開發、前端開發、美術設計、文件撰寫以及報告。為了更有效地進行專案管理，這五大項目進一步細分成多個小項目。我們採取專業分工的策略，根據每位組員的專長和強項來分配相應的工作，以確保每個階段的工作都能高效且專業地完成。不僅提高了工作效率，也讓本組成員能在其擅長的領域內發揮最大的潛力。

* 以下為Github各組員 commit的次數：



▲圖 4-2‑1 Github頁面

# 需求模型

## 使用者需求

▼表 5-1‑1使用者需求表

|  |  |
| --- | --- |
| Use Case | 需求 |
| 登入與註冊 | 使用者透過電子郵件註冊及登入App |
| 房源篩選與搜索 | 使用者可以透過篩選器或搜尋功能找到適合的房源 |
| 訂閱功能 | 使用者可以依照自己的喜好選擇標籤，當有符合標籤的房源時，系統會自動推送 |
| 瀏覽房屋資訊 | 使用者點擊房屋資訊卡片查看詳細的房屋資訊 |
| 使用Line智能助手-規劃中 | 使用者可使用Line聊天機器人，隨時提問有關租屋知識或是房源的相關問題 |
| 收藏房源 | 使用者可收藏感興趣的房源並在需要時查看 |
| 瀏覽歷史紀錄 | 使用者可隨時查看曾經瀏覽過的房源 |
| 房源管理 | 出租者可透過平台管理房源、更新出租資料 |

## 使用個案圖



▲圖 5-2‑1使用個案圖

## 使用個案描述

▼表 5-3‑1房屋查找

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：房屋查找(篩選、搜尋) | | |
| 行為者 | 使用者 | |
| 前提 | 使用者進入到「搜尋」頁面中 | |
| 結束狀態 | 使用者可找到符合條件的房屋訊息 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.點選篩選和搜尋頁面 | | 2.顯示篩選或搜尋器 |
| 3.選擇篩選條件或搜尋關鍵字 | | 4.顯示符合篩選或搜尋的房源 |

▼表 5-3‑2訂閱功能

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：訂閱功能 | | |
| 行為者 | 使用者 | |
| 前提 | 使用者已登入 | |
| 結束狀態 | 使用者選擇完訂閱標籤並看到已訂閱的房源 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.訂閱有興趣之標籤 | | 2.保存使用者訂閱標籤 |
|  | | 3.顯示符合訂閱標籤之房源 |
|  | | 4.新增符合標籤房源時，推播消息給使用者 |
| 5.接收到符合訂閱標籤的房源 | |  |

▼表 5-3‑3瀏覽房屋詳細資訊

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：瀏覽房屋詳細資訊 | | |
| 行為者 | 使用者 | |
| 前提 | 使用者點擊房屋資訊卡片 | |
| 結束狀態 | 使用者瀏覽房源詳細資訊 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.在相關頁面篩選搜尋查看房源 | |  |
| 2.點擊有興趣之房源資訊卡片 | | 3.顯示房屋詳細資訊頁面 |

▼表 5-3‑4使用Line智能助手-規劃中

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：使用Line智能助手-規劃中 | | |
| 行為者 | 使用者 | |
| 前提 | 使用者進入APP加入Line官方帳號 | |
| 結束狀態 | 使用者使用Line智能助手獲得解答 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.在APP中點選智能助手，且加入官方Line | |  |
| 2.提問房源相關問題，可點選罐頭問題，或自行輸入 | | 3.解決使用者提問之問題 |

▼表 5-3‑5收藏房源

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：收藏房源 | | |
| 行為者 | 使用者 | |
| 前提 | 使用者已登入 | |
| 結束狀態 | 把有興趣之房源進行收藏，且可在收藏頁面查看 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.在首頁或個人頁面查看房屋資訊卡片 | |  |
| 2.點擊房屋資訊卡片右上角或房源詳細資訊頁面右上角皆可收藏房源 | | 3.將使用者收藏之房源進行儲存 |
| 4.可在個人頁面查看或管理收藏內容 | |  |

▼表 5-3‑6瀏覽歷史紀錄

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：瀏覽歷史紀錄 | | |
| 行為者 | 使用者 | |
| 前提 | 使用者已登入，且點擊過房屋資訊卡片 | |
| 結束狀態 | 瀏覽已點擊過的房屋資訊卡片紀錄，並查看詳細資訊 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.點擊個人頁面”歷史紀錄” | | 2.顯示使用者曾經點擊過的房屋資訊卡片 |
| 3.點擊想查看之房源 | | 4.顯示房屋詳細資訊頁面 |

▼表 5-3‑7房源管理(新增)

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：房源管理(新增) | | |
| 行為者 | 使用者 | |
| 前提 | 使用者已登入 | |
| 結束狀態 | 於刊登頁面成功新增房源 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.選擇刊登頁面 | | 2.顯示所有已刊登的房源，如未刊登過則空白 |
| 3.點擊右下角+號以新增房源 | |  |
| 4.填寫房屋相關資訊 | | 5.顯示房屋詳細資訊預覽畫面 |
| 6.確認房屋詳細資訊預覽畫面 | | 7.使用者確認完成後儲存新增的房源 |
| 8.新增的房源會位於刊登頁面最上方 | |  |

▼表 5-3‑8房源管理(編輯)

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：房源管理(編輯) | | |
| 行為者 | 使用者 | |
| 前提 | 使用者已登入，且有已刊登房源 | |
| 結束狀態 | 於刊登頁面成功新增編輯後的房源 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
| 1.選擇刊登頁面 | | 2.顯示所有已刊登的房源 |
| 3.點擊房屋資訊卡片，修改資訊 | | 4.顯示房屋詳細資訊預覽畫面 |
| 5.確認房屋詳細資訊預覽畫面 | | 6.使用者確認完成後儲存編輯後的房源 |
| 7.編輯後的房源會位於刊登頁面最上方 | |  |

▼表 5-3‑9推播訂閱資訊

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：推播訂閱資訊 | | |
| 行為者 | 系統 | |
| 前提 | 符合使用者訂閱標籤且使用者開啟推播功能，並有新的符合房源上架 | |
| 結束狀態 | 推播符合訂閱的房源給使用者 | |
| 事件路徑： | | |
| Actor 動作 | | 系統回應 |
|  | | 1.根據訂閱標籤，推播給使用者 |
| 2.成功接收推播內容 | |  |

## 分析類別圖



▲圖 5-4‑1分析類別圖

# 設計模型

## 循序圖(Sequential diagram)或通訊圖(Communication diagram)



▲圖 6-1‑1循序圖－註冊會員



▲圖 6-1‑2循序圖-登入



▲圖 6-1‑3循序圖-更改密碼



▲圖 6-1‑4循序圖-房屋查找



▲圖 6-1‑5循序圖-訂閱



▲圖 6-1‑6循序圖-修改個人資料



▲圖 6-1‑7循序圖-收藏房屋



▲圖 6-1‑8循序圖-編輯我的收藏



▲圖 6-1‑9循序圖-瀏覽我的收藏



▲圖 6-1‑10循序圖-查看瀏覽紀錄

## 設計類別圖



▲圖 6-2‑1設計類別圖

# 實作模型

## 佈署圖(Deployment diagram)



▲圖 7-1‑1佈署圖

## 套件圖(Package diagram)

|  |  |
| --- | --- |
| ▲圖 7-2‑1套件圖－爬蟲Get detail | ▲圖 7-2‑2套件圖－爬蟲Get image |
| ▲圖 7-2‑3套件圖－爬蟲Get URL    ▲圖 7-2‑4套件圖－串接前後端Flask API | ▲圖 7-2‑5套件圖－Line message API    ▲圖 7-2‑6套件圖－後端 重複資料判斷 |
| ▲圖 7-2‑7套件圖－資料存入MySQL | ▲圖 7-2‑8套件圖－圖片存入MySQL |

## 元件圖(Component diagram)



▲圖 7-3‑1元件圖

## 狀態機(State machine)



▲圖 7-4‑1狀態機－登入



▲圖 7-4‑2狀態機－註冊



▲圖 7-4‑3狀態機－忘記密碼



▲圖 7-4‑4狀態機－修改密碼



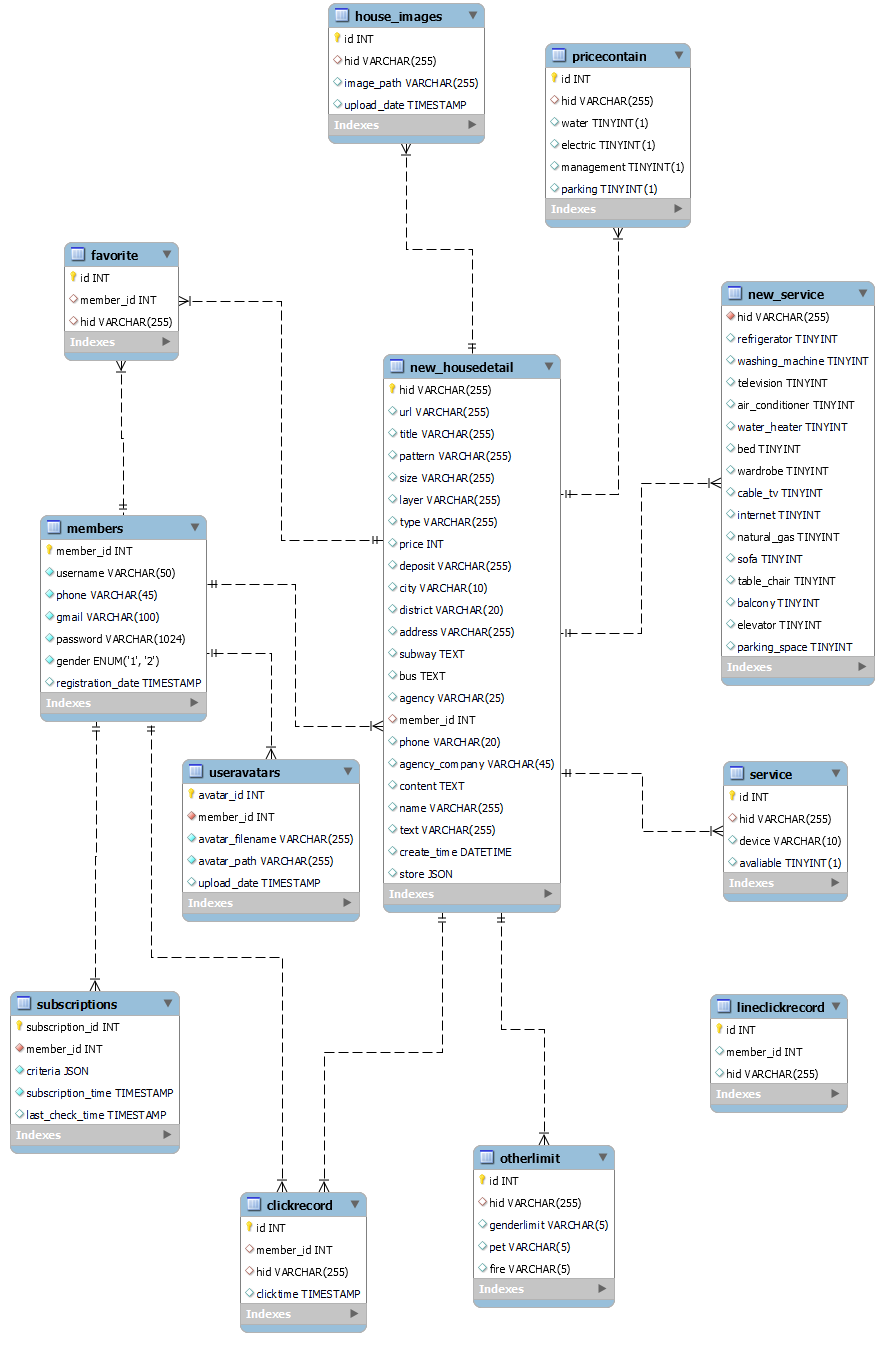
▲圖 7-4‑5狀態-編輯訂閱標籤



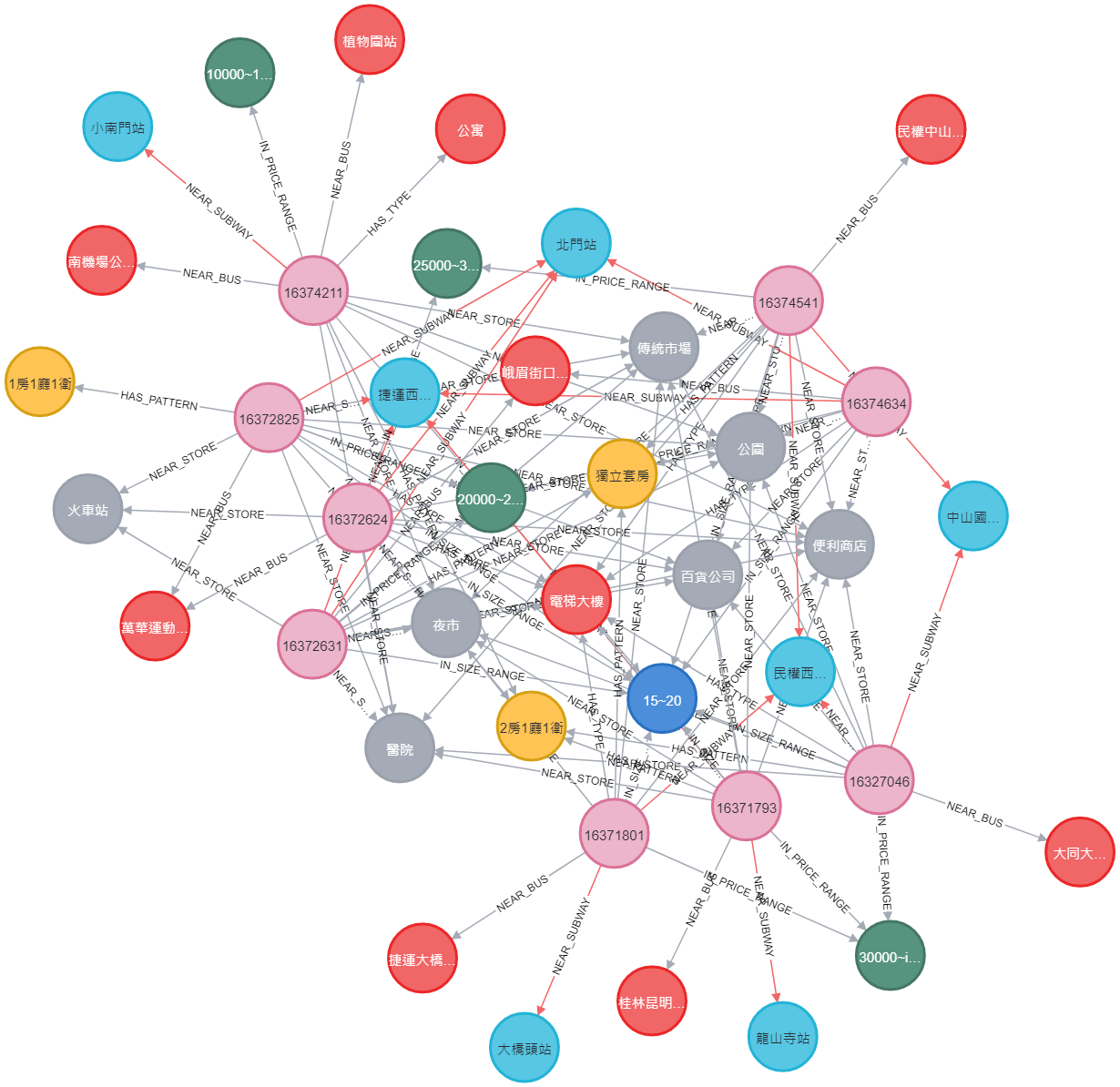
▲圖 7-4‑6狀態機－搜尋房源

# 資料庫設計

## 資料庫關聯圖



▲圖 8-1‑1 MySQL資料庫ER圖



▲圖 8-1‑2 Neo4j圖形式資料庫

為了實現本組「過濾重複物件」以及「LINE智能助手」等功能，創建neo4j圖形結構的資料庫建立知識圖譜，更自然地表示各房屋物件之間的關聯，以節點之間的關聯路徑做比較，更準確的篩選出相同房屋物件，以及精確的回答使用者向智能助手提出的問題，除了實現本組目標功能也達到縮短搜尋時間的目的，更利於前後端做串連以及方便使用者做使用

## 表格及其Meta data

new\_housedetail用於紀錄每筆房屋物件的所有資訊，包括：房型、坪數、物件名稱等等。service記錄房屋有提供的附屬設備。address為房屋所在地址。agency為負責該房屋銷售的仲介人士及仲介公司。useravatars為使用者的頭貼，members為會員的所有資料，包括：username、password等等。

▼表 8-2‑1資料表

|  |  |  |
| --- | --- | --- |
| 資料表編號 | 資料表英文名稱 | 資料表中文名稱 |
| 01 | new\_housedetail | 房屋詳細資料 |
| 02 | service | 房屋設備 |
| 03 | address | 房屋地址 |
| 04 | agency | 房屋仲介 |
| 05 | members | 會員資料 |
| 06 | useravatars | 使用者頭貼 |

01 new\_housedetail(房屋詳細資料)

▼表 8-2‑2資料表描述－01房屋詳細資料

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 中文名稱 | 房屋詳細資料 | | 資料表編號 | | 01 | | |
| 英文名稱 | New\_housedetail | | 主索引 | | hid | | |
| 資料檔陳述 | 紀錄每一筆房屋資料的細節 | | | | | | |
| 欄位名稱 | 資料型態 | 長度 | 允許空值 | 中文 | | 備註 | 預設值 |
| hid | VARCHAR | 255 | 否 | 房屋編號 | | PK |  |
| url | VARCHAR | 255 |  | 連結網址 | |  |  |
| title | VARCHAR | 255 |  | 標題 | |  |  |
| pattern | VARCHAR | 255 |  | 房型 | |  |  |
| size | VARCHAR | 255 |  | 大小 | |  |  |
| layer | VARCHAR | 255 |  | 樓層 | |  |  |
| type | VARCHAR | 255 |  | 建築物特徵 | |  |  |
| price | INT |  |  | 價格 | |  |  |
| deposit | VARCHAR | 255 |  | 押金 | |  |  |
| address | VARCHAR | 255 |  | 地址 | |  |  |
| subway | TEXT |  |  | 附近捷運站 | |  |  |
| bus | TEXT |  |  | 附近公車站 | |  |  |
| agency\_id | INT |  |  | 仲介編號 | |  |  |
| agency | VARCHAR | 25 |  | 仲介 | |  |  |
| agency\_company | VARCHAR | 45 |  | 仲介公司 | |  |  |
| content | TEXT |  |  | 介紹文 | |  |  |
| name | VARCHAR | 255 |  |  | |  |  |
| text | VARCHAR | 255 |  |  | |  |  |

02 service(房屋設備)

▼表 8-2‑3資料表描述－02房屋設備

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 中文名稱 | 房屋設備 | | 資料表編號 | | 02 | | |
| 英文名稱 | service | | 主索引 | | id | | |
| 資料檔陳述 | 房屋提供的附屬設備 | | | | | | |
| 欄位名稱 | 資料型態 | 長度 | 允許空值 | 中文 | | 備註 | 預設值 |
| id | INT |  | 否 | 編號 | | PK、AI |  |
| hid | VARCHAR | 255 |  | 房屋編號 | |  |  |
| device | VARCHAR | 10 |  | 設備 | |  |  |
| available | TINYINT | 1 |  | 是否可用 | |  |  |

03 address(房屋地址)

▼表 8-2‑4資料表描述－03房屋地址

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 中文名稱 | 房屋地址 | | 資料表編號 | | 03 | | |
| 英文名稱 | address | | 主索引 | | address\_id | | |
| 資料檔陳述 | 紀錄房屋地址 | | | | | | |
| 欄位名稱 | 資料型態 | 長度 | 允許空值 | 中文 | | 備註 | 預設值 |
| address\_id | INT |  | 否 | 地址編號 | | PK、AI |  |
| hid | VARCHAR | 255 | 否 | 房屋編號 | |  |  |
| title | VARCHAR | 255 |  | 標題 | |  |  |
| address | VARCHAR | 255 |  | 地址 | |  |  |

04 agency(房屋仲介)

▼表 8-2‑5資料表描述－04房屋仲介

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 中文名稱 | 房屋仲介 | | 資料表編號 | | 04 | | |
| 英文名稱 | agency | | 主索引 | | agency\_id | | |
| 資料檔陳述 | 特定房屋所屬仲介 | | | | | | |
| 欄位名稱 | 資料型態 | 長度 | 允許空值 | 中文 | | 備註 | 預設值 |
| agency\_id | INT |  | 否 | 仲介編號 | | PK、AI |  |
| agency | VARCHAR | 25 |  | 仲介 | |  |  |
| agency\_company | VARCHAR | 45 |  | 仲介公司 | |  |  |

05 members(會員資料)

▼表 8-2‑6資料表描述－05會員資料

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 中文名稱 | 會員資料 | | 資料表編號 | | 05 | | |
| 英文名稱 | members | | 主索引 | | member\_id | | |
| 資料檔陳述 | 會員詳細資料 | | | | | | |
| 欄位名稱 | 資料型態 | 長度 | 允許空值 | 中文 | | 備註 | 預設值 |
| member\_id | INT |  | 否 | 會員編號 | | PK、AI |  |
| username | VARCHAR | 50 | 否 | 會員名稱 | |  |  |
| phonenumber | VARCHAR | 45 | 否 | 手機號碼 | |  |  |
| email | VARCHAR | 100 | 否 | 電子郵件 | |  |  |
| password | VARCHAR | 1024 | 否 | 密碼 | |  |  |
| gender | ENUM |  | 否 | 性別 | |  |  |
| registration\_date | TIMESTAMP |  |  | 註冊日期 | |  |  |

06 useravatars(使用者頭貼)

▼表 8-2‑7資料表描述－06使用者頭貼

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 中文名稱 | 使用者頭貼 | | 資料表編號 | | 06 | | |
| 英文名稱 | useravatars | | 主索引 | | avatar\_id | | |
| 資料檔陳述 | 紀錄每一位使用者的頭貼 | | | | | | |
| 欄位名稱 | 資料型態 | 長度 | 允許空值 | 中文 | | 備註 | 預設值 |
| avatar\_id | INT |  | 否 | 頭貼編號 | | PK、AI |  |
| member\_id | INT |  | 否 | 會員編號 | |  |  |
| avatar\_filename | VARCHAR | 255 | 否 | 頭貼名稱 | |  |  |
| avatar\_path | VARCHAR | 255 | 否 | 頭貼路徑 | |  |  |
| upload\_date | TIMESTAMP |  |  | 上傳日期 | |  |  |

# 程式

## 元件清單及其規格描述

▼表 9-1‑1元件清單與規格描述表(前端)-App介面編寫

|  |  |  |
| --- | --- | --- |
| Dart檔案(前端)-App介面編寫 | | |
| 編號 | 檔案名稱 | 功能 |
| 1-1-1 |  |  |
| 1-1-2 |  |  |
| 1-1-3 |  |  |
| 1-1-4 |  |  |
| 1-1-5 |  |  |
| 1-1-6 |  |  |
| 1-1-7 |  |  |

▼表 9-1‑2元件清單與規格描述表(後端)-爬蟲

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－爬蟲 | | |
| 編號 | 檔案名稱 | 功能 |
| 2-1-1 | 591crawler1.py | 爬取房屋個別物件鏈結網址(url) |
| 2-1-2 | 591crawler2.py  591crawler3.py | 爬取房屋圖片 |
| 2-1-3 | 591crawler4.py | 爬取房屋詳細資訊 |

表 9-1‑3元件清單與規格描述表(後端)-資料庫(MySQL)

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－資料庫(MySQL) | | |
| 編號 | 檔案名稱 | 功能 |
| 2-2-1 | Mysql\_import\_house\_data.py | 將經過整理的房屋資料轉換成對應的格式存入資料庫(MySQL) |

表 9-1‑4元件清單與規格描述表(後端)-圖形資料庫(Neo4j)

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－圖形資料庫(Neo4j) | | |
| 編號 | 檔案名稱 | 功能 |
| 2-3-1 | neo4j\_transfer\_and\_creation.py | 將mysql資料導入neo4j，並建立節點(房屋物件、價格、房屋類型、房間類型、坪數)以及建立關聯 |
| 2-3-2 | neo4j\_nearby\_features.py | 將整理好的.json(附近特色:學校、商店、醫院……)導入先前建好的neo4j資料庫，建立節點及關聯 |
| 2-3-3 | neo4j\_transportation.py | 將整理好的.json(附近交通:公車、捷運)導入先前建好的neo4j資料庫，建立節點及關聯 |

表 9-1‑5元件清單及其規格描述表(後端)-Flask API

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－Flask API | | |
| 編號 | 檔案名稱 | 功能 |
| 2-4-1 |  |  |
| 2-4-2 |  |  |
| 2-4-3 |  |  |
| 2-4-4 |  |  |
| 2-4-5 |  |  |

表 9-1‑6元件清單及其規格描述表(後端)-ChatGPT API

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－ChatGPT API | | |
| 編號 | 檔案名稱 | 功能 |
| 2-5-1 | house\_describe\_with\_neo4j.py  house\_describe\_with\_neo4j.ipynb | 連接neo4j給chatgpt當作回答的依據。 |
| 2-5-2 | chatgpt\_question.py  chatgpt\_question. ipynb | 連接chatgpt，且提供neo4j的房屋資料當作回答的參考依據。 |
| 2-5-3 | Line\_all\_function.py | Line聊天機器人連接到chatgpt做智慧問答。 |

表 9-1‑7元件清單及其規格描述表(後端)-AI

|  |  |  |
| --- | --- | --- |
| Python檔案(後端)－AI | | |
| 編號 | 檔案名稱 | 功能 |
| 2-6-1 | house\_similarity.py  house\_detail.py | 房屋的地址、坪數、設施等文字性相似度比較。 |
| 2-6-2 | Yolo.py  Yolo+clip.py | 房屋圖片的物件偵測與相似度比較。 |
| 2-6-3 | ner\_clean\_1.py  nearby\_ORG\_cleaned\_2.json | 提取房屋描述內有關附近生活機能的店家或設施。 |
| 2-6-4 | subway\_bus\_ckip.py  subway\_and\_bus.json | 提取房屋附近的捷運站及公車站，供neo4j資料庫使用。 |

## 其他附屬之各種元件

表 9-2‑1 虛擬碼-Mysql\_import\_house\_data.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-2-1 | 程式名稱 | Mysql\_import\_house\_data.py |
| 目的 | 將經過整理的房屋資料轉換成對應的格式存入資料庫(MySQL) | | |
| 虛擬碼 | | | |
| 1. 初始化與設置    * 載入所需的模組和庫：      + mysql.connector：用於連接和操作 MySQL 資料庫。      + json：讀取和解析 JSON 文件。 2. 讀取 JSON 檔案    * 打開並讀取 JSON 檔案 detail.json，將其內容解析為 Python 字典 data。 3. 連接 MySQL 資料庫    * 使用指定的主機、用戶名、密碼和資料庫名稱連接到 MySQL 資料庫。    * 創建游標 cursor 來執行 SQL 查詢。 4. 創建 ghdetail 資料庫    * 如果資料庫 ghdetail 不存在，則創建該資料庫。 5. 選擇並使用 ghdetail 資料庫    * 使用 USE ghdetail 命令選擇並使用該資料庫。 6. 清空並創建 housedetail 表格    * 如果 housedetail 表格已存在，則清空該表格。    * 如果 housedetail 表格不存在，則創建表格，其中包含房屋詳細資訊的各個字段。 7. 檢查並創建 service 表格    * 如果 service 表格不存在，則創建表格，其中包含設備資訊及其可用性，並設置外鍵 hid 來引用 housedetail 表格。 8. 刪除 service 表格中的所有數據    * 清空 service 表格中的所有現有數據。 9. 將 JSON 檔案的數據插入到 housedetail 和 service 表格中    * 對於每個 JSON 項目 item：      + 提取房屋 ID (hid)、網址 (url)、房屋資訊 (houseinfo)、地理位置 (positionround)、服務清單 (servicelist) 和備註 (remark)。      + 將地鐵站和公車站的資訊合併為字符串，並將備註內容合併為單一字符串。      + 將提取的數據插入到 housedetail 表格中。      + 將每個服務項目的設備名稱和可用性狀態插入到 service 表格中。 10. 確認提交資料庫變更     * 使用 conn.commit() 確認並提交所有變更到資料庫。 11. 關閉游標和連接     * 關閉資料庫游標 cursor 和連接 conn。 12. 輸出成功消息     * 輸出成功將資料匯入 MySQL 資料庫的消息。 | | | |

表 9-2‑2虛擬碼-neo4j\_transfer\_and\_creation.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-3-1 | 程式名稱 | neo4j\_transfer\_and\_creation.py |
| 目的 | 將mysql資料導入neo4j，並建立節點(房屋物件、價格、房屋類型、房間類型、坪數)以及建立關聯 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置    * 載入所需的模組和庫：      + mysql.connector.connect 和 mysql.connector.Error：用於連接 MySQL 數據庫和處理錯誤。      + py2neo.Graph, py2neo.Node, py2neo.Relationship：用於連接 Neo4j 數據庫，創建節點和關係。      + time：用於在重試連接 Neo4j 時進行延遲。      + re：用於正則表達式處理文本。 2. 連接到 MySQL 數據庫    * 嘗試連接到 MySQL 數據庫，並處理可能的連接錯誤。 3. 定義函數 connect\_to\_neo4j    * 輸入參數: retries（重試次數）和 wait（等待時間）。    * 在指定的重試次數內嘗試連接到 Neo4j，如果失敗，則等待一段時間再重試。    * 成功後返回 Neo4j 的 graph 對象；如果所有嘗試均失敗，則退出程式。 4. 連接到 Neo4j 數據庫    * 使用 connect\_to\_neo4j 函數嘗試連接到 Neo4j。 5. 查詢 MySQL 數據    * 使用 SQL 語句從 MySQL 數據庫中提取所需數據。    * 處理可能的查詢錯誤。 6. 定義價格和大小範圍    * 定義價格範圍 price\_ranges 和大小範圍 size\_ranges。 7. 定義函數 parse\_size    * 輸入參數: size\_str（大小字符串）。    * 使用正則表達式從字符串中提取數字並返回；如果提取失敗，則返回 None。 8. 處理數據並寫入 Neo4j    * 對於每個從 MySQL 提取的數據行 row：      + 提取並解析屬性信息（如 hid, pattern, size\_str, layer, type, price, address, subway, bus）。      + 如果大小無法解析，則跳過該屬性。      + 創建房屋節點：使用 Node 創建包含屬性信息的房屋節點，並將其寫入 Neo4j。      + 創建物件類型節點：創建或合併物件類型節點到 Neo4j。      + 創建物件類型（pattern）節點：創建或合併物件類型（pattern）節點到 Neo4j。      + 創建價格範圍節點：根據價格，創建或合併相應的價格範圍節點到 Neo4j，並創建與房屋節點的關係。      + 創建大小範圍節點：根據大小，創建或合併相應的大小範圍節點到 Neo4j，並創建與房屋節點的關係。      + 創建類型和 pattern 關係：為房屋節點創建與類型和 pattern 節點的關係。    * 處理可能的寫入錯誤。 9. 關閉 MySQL 連接    * 關閉 MySQL 游標和連接。 | | | |

表 9-2‑3虛擬碼-neo4j\_nearby\_features.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-3-2 | 程式名稱 | neo4j\_nearby\_features.py |
| 目的 | 將整理好的.json(附近特色:學校、商店、醫院……)導入先前建好的neo4j資料庫，建立節點及關聯 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫：   + json：讀取和解析 JSON 文件。   + py2neo.Graph, py2neo.Node, py2neo.Relationship：用於連接 Neo4j 數據庫，創建節點和關係。   + time：用於在重試連接 Neo4j 時進行延遲。   + os：用於檢查文件是否存在。  1. 定義函數 connect\_to\_neo4j  * 輸入參數: retries（重試次數）和 wait（等待時間）。 * 在指定的重試次數內嘗試連接到 Neo4j，如果失敗，則等待一段時間再重試。 * 成功後返回 Neo4j 的 graph 對象；如果所有嘗試均失敗，則退出程式。  1. 連接到 Neo4j 數據庫  * 使用 connect\_to\_neo4j 函數嘗試連接到 Neo4j。  1. 設定 JSON 文件路徑  * 指定 JSON 文件的路徑 json\_file\_path。  1. 檢查文件是否存在  * 使用 os.path.exists 函數檢查指定的 JSON 文件是否存在。 * 如果文件不存在，則輸出錯誤消息並退出程式。  1. 讀取 JSON 文件  * 使用 open 函數和 json.load 讀取並解析 JSON 文件的內容。 * 處理可能的 JSON 解碼錯誤。  1. 處理數據並寫入 Neo4j  * 對於 JSON 數據中的每個項目 entry：   + 提取房屋 ID (hid) 和商店列表 (stores)。   + 檢查 stores 是否為列表類型；如果不是，則跳過該項目。   + 查找對應的 Property 節點：根據 hid 在 Neo4j 中查找對應的 Property 節點。   + 如果找不到對應的 Property 節點，則跳過該項目。   + 創建或獲取 Store 節點並創建關係：     - 對於每個商店名稱 store，創建或合併對應的 Store 節點。     - 創建 Property與 Store 節點之間的 NEAR\_STORE 關係。 * 處理可能的數據處理錯誤。  1. 成功完成數據整合  * 輸出數據成功整合的消息。 | | | |

表 9-2‑4虛擬碼-neo4j\_transportation.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-3-3 | 程式名稱 | neo4j\_transportation.py |
| 目的 | 將整理好的.json(附近交通:公車、捷運)導入先前建好的neo4j資料庫，建立節點及關聯 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫：   + json：讀取和解析 JSON 文件。   + py2neo.Graph, py2neo.Node, py2neo.Relationship：用於連接 Neo4j 數據庫，創建節點和關係。   + time：用於在重試連接 Neo4j 時進行延遲。   + os：用於檢查文件是否存在。  1. 定義函數 connect\_to\_neo4j  * 輸入參數: retries（重試次數）和 wait（等待時間）。 * 在指定的重試次數內嘗試連接到 Neo4j，如果失敗，則等待一段時間再重試。 * 成功後返回 Neo4j 的 graph 對象；如果所有嘗試均失敗，則退出程式。  1. 連接到 Neo4j 數據庫  * 使用 connect\_to\_neo4j 函數嘗試連接到 Neo4j。  1. 設定 JSON 文件路徑  * 指定 JSON 文件的路徑 json\_file\_path。  1. 檢查文件是否存在  * 使用 os.path.exists 函數檢查指定的 JSON 文件是否存在。 * 如果文件不存在，則輸出錯誤消息並退出程式。  1. 讀取 JSON 文件  * 使用 open 函數和 json.load 讀取並解析 JSON 文件的內容。  1. 處理數據並寫入 Neo4j  * 對於 JSON 數據中的每個項目 entry：   + 提取房屋 ID (hid)、地鐵站列表 (subways) 和公車站列表 (buses)。   + 查找對應的 Property 節點：根據 hid 在 Neo4j 中查找對應的 Property 節點。   + 如果找不到對應的 Property 節點，則跳過該項目。   + 創建或獲取 Subway 節點並創建關係：     - 對於每個地鐵站名稱 subway\_station，創建或合併對應的 Subway 節點到 Neo4j。     - 創建 Property 節點與 Subway 節點之間的 NEAR\_SUBWAY 關係。   + 創建或獲取 Bus 節點並創建關係：     - 對於每個公車站名稱 bus\_stop，創建或合併對應的 Bus 節點到 Neo4j。     - 創建 Property 節點與 Bus 節點之間的 NEAR\_BUS 關係。 * 處理可能的數據處理錯誤。  1. 成功完成數據整合  * 輸出數據成功整合的消息。 | | | |

▼表 9-2‑5虛擬碼-house\_describe\_with\_neo4j.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-5-1 | 程式名稱 | house\_describe\_with\_neo4j.py |
| 目的 | 連接neo4j給chatgpt當作回答的依據。 | | |
| 虛擬碼 | | | |
| 1.連接到 Neo4j 資料庫   * 使用 py2neo 模組連接到 Neo4j 資料庫，指定連接的 URI 和授權憑據。   2.定義函數 generate\_description   * 輸入參數：hid（房屋 ID） * 建立查詢語句：   + 匹配房屋節點 h，該節點的 hid 等於輸入參數 hid。   + 可選匹配房屋節點 h 與店家節點 s 之間的 NEAR\_STORE 關係。   + 返回房屋節點 h 和所有匹配的店家節點 s（以列表形式返回）。 * 執行查詢，將結果存儲到 result。   3.檢查查詢結果   * 如果 result 為空：   + 返回 "未找到與該HID相關的房屋資料。" * 否則：   + 獲取房屋資訊 house\_info 和店家資訊 stores。   4.初始化描述列表   * 創建一個空的描述列表 descriptions。   5.生成房屋屬性描述   * 如果 house\_info 包含 address：   + 添加描述 "房屋地址位於 {address}。" * 如果 house\_info 包含 type：   + 添加描述 "該房屋類型為 {type}。" * 如果 house\_info 包含 pattern：   + 添加描述 "房屋格局為 {pattern}。" * 如果 house\_info 包含 size：   + 添加描述 "房屋大小為 {size} 坪。" * 如果 house\_info 包含 layer：   + 添加描述 "樓層為 {layer}。" * 如果 house\_info 包含 price：   + 添加描述 "房屋租金為 {price} 元。" * 如果 house\_info 包含 subway：   + 添加描述 "距離最近的地鐵站為 {subway}。" * 如果 house\_info 包含 bus：   + 添加描述 "附近的公車站包括 {bus}。"   6.生成店家描述   * 如果 stores 不為空：   + 從 stores 中提取每個店家的 name。   + 如果 store\_names 不為空：     - 添加描述 "附近有以下店家：{店家清單}。"   7.返回完整描述   * 將所有描述合併成一個字符串並返回。   8.主程式   * 獲取用戶輸入的 hid。 * 調用 generate\_description 函數，將 hid 傳入。 * 打印生成的描述。 | | | |

表 9-2‑6虛擬碼- chatgpt\_question.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-5-2 | 程式名稱 | chatgpt\_question.py |
| 目的 | 連接chatgpt，且提供neo4j的房屋資料當作回答的參考依據。 | | |
| 虛擬碼 | | | |
| 1.**初始化與設置**   * 設定 OpenAI 的 API 密鑰。 * 使用 py2neo 模組連接到 Neo4j 資料庫，指定連接的 URI 和授權憑據。   2.**定義函數 generate\_description**   * **輸入參數**：hid（房屋 ID） * **建立查詢語句**：   + 匹配房屋節點 h，該節點的 hid 等於輸入參數 hid。   + 可選匹配房屋節點 h 與店家節點 s 之間的 NEAR\_STORE 關係。   + 返回房屋節點 h 和所有匹配的店家節點 s（以列表形式返回）。 * **執行查詢**，將結果存儲到 result。   3.**檢查查詢結果**   * 如果 result 為空：   + 返回 "未找到與該HID相關的房屋資料。" * 否則：   + 獲取房屋資訊 house\_info 和店家資訊 stores。   4.**初始化描述列表**   * 創建一個空的描述列表 descriptions。   5.**生成房屋屬性描述**   * 如果 house\_info 包含 address：   + 添加描述 "房屋地址位於 {address}。" * 如果 house\_info 包含 type：   + 添加描述 "該房屋類型為 {type}。" * 如果 house\_info 包含 pattern：   + 添加描述 "房屋格局為 {pattern}。" * 如果 house\_info 包含 size：   + 添加描述 "房屋大小為 {size} 坪。" * 如果 house\_info 包含 layer：   + 添加描述 "樓層為 {layer}。" * 如果 house\_info 包含 price：   + 添加描述 "房屋租金為 {price} 元。" * 如果 house\_info 包含 subway：   + 添加描述 "距離最近的捷運站為 {subway}。" * 如果 house\_info 包含 bus：   + 添加描述 "附近的公車站包括 {bus}。"   6.**生成店家描述**   * 如果 stores 不為空：   + 從 stores 中提取每個店家的 name。   + 如果 store\_names 不為空：     - 添加描述 "附近有以下店家：{店家清單}。"   7.**返回完整描述**   * 將所有描述合併成一個字符串並返回。   8.**定義函數 gpt\_analyze\_input**   * **輸入參數**：hid（房屋 ID），message（使用者提問） * 調用 generate\_description 函數以生成房屋描述。 * 如果描述中包含 "未找到與該HID相關的房屋資料"：   + 返回該訊息。 * 否則，構建 OpenAI GPT 的 prompt，包含房屋描述和使用者提問。 * 使用 OpenAI GPT 生成回答：   + 發送 prompt 給 OpenAI GPT-3.5-turbo-16k 模型。   + 使用系統訊息設定模型角色為專業房屋中介助手，並指定回答語言為繁體中文。 * 返回模型生成的回覆。   9.**例外處理**   * 如果調用 OpenAI API 或其他操作發生異常，捕捉異常並打印錯誤訊息。 * 返回 "在處理您的請求時發生錯誤。"   10.**主程式**   * 獲取使用者輸入的 hid 和 question。 * 打印輸入訊息。 * 調用 gpt\_analyze\_input 函數，傳入 hid 和 question。 * 打印生成的回答。 * 如果未收到回答，則打印錯誤訊息。 | | | |

表 9-2‑7虛擬碼- Line\_all\_function.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-5-3 | 程式名稱 | Line\_all\_function.py |
| 目的 | Line聊天機器人連接到chatgpt做智慧問答。 | | |
| 虛擬碼 | | | |
| 1. 連接到 Neo4j 資料庫    * 使用 py2neo 模組連接到 Neo4j 資料庫，指定連接的 URI (bolt://localhost:7687) 和授權憑據（使用者名稱和密碼）。 2. 設置 Line Bot API 和 Webhook Handler    * 初始化 LineBotApi 和 WebhookHandler，使用各自的金鑰和通行碼進行身份驗證。 3. 啟用 ngrok 並創建公共 URL    * 使用 ngrok 來建立本地伺服器的公共 URL。    * 設置 ngrok 認證令牌。    * 斷開任何現有的隧道連接。    * 建立新隧道並打印公共 URL。 4. 初始化 Flask 應用    * 創建 Flask 應用實例，並設置隨機生成的秘密金鑰。 5. 定義全局字典用於存儲使用者狀態    * 創建一個全局字典 user\_states 用於儲存使用者狀態。 6. 定義函數 generate\_description    * 輸入參數：hid（房屋 ID）    * 查詢語句：      + 匹配房屋節點 h，該節點的 hid 等於輸入參數 hid。      + 可選匹配房屋節點 h 與店家節點 s 之間的 NEAR\_STORE 關係。      + 返回房屋節點 h 和所有匹配的店家節點 s（以列表形式返回）。    * 執行查詢：將結果存儲到 result。    * 生成描述：      + 檢查 result 是否為空，如果是，返回"未找到與該HID相關的房屋資料"。      + 從 result 提取房屋信息和店家列表。      + 根據房屋信息生成描述，並將描述存入列表 descriptions。      + 返回連接所有描述的字符串。 7. 定義函數 gpt\_analyze\_input    * 輸入參數：message（使用者消息），user\_id（使用者 ID）    * 專業系統消息：設置系統消息，定義 GPT 回應的範圍和語言風格。    * 判斷使用者消息類型：      + 如果消息包含 "租屋知識" 或 "租屋相關"，則生成對應的 prompt 用於租屋知識查詢。      + 否則，提取 hid 並生成房屋描述，生成對應的 prompt 用於房屋信息查詢。    * 調用 OpenAI API：      + 使用 openai.ChatCompletion.create 調用 OpenAI API，傳入系統消息和 prompt 生成回應。      + 將生成的回應提取並返回給使用者。 8. 設置 Linebot 回調函數 callback    * 獲取簽名和請求體：      + 從請求頭獲取 X-Line-Signature。      + 以文本形式獲取請求體。    * 處理請求：使用 WebhookHandler 處理請求，並返回 'OK'。    * 處理異常：如果簽名無效，則返回 400 錯誤。 9. 定義函數 handle\_message    * 獲取使用者 ID 和消息：從事件中提取使用者 ID 和消息。    * 檢查消息格式：      + 使用正則表達式檢查消息是否符合格式：member\_id: <數字> hid: <數字>。      + 如果格式正確，更新使用者狀態並回應驗證成功。      + 否則，檢查使用者狀態是否已驗證，如果已驗證則分析輸入，否則提示重新驗證。 10. 啟動 Flask 應用  * 使用 app.run() 啟動 Flask 應用。 | | | |

表 9-2‑8虛擬碼-house\_detail.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-6-1 | 程式名稱 | house\_detail.py |
| 目的 | 房屋的地址、坪數、設施等文字性相似度比較。 | | |
| 虛擬碼 | | | |
| 1.初始化與設置   * 載入所需的模組和庫：   + json 用於讀取 JSON 文件。   + torch 和 torch.nn.functional 用於深度學習操作和計算餘弦相似度。   + ckiptagger.WS 用於中文分詞。   + transformers.BertTokenizer 和 transformers.BertModel 用於 BERT 中文模型的分詞與嵌入。   + re 用於正則表達式處理文本。 * 初始化 CKIP 的斷詞模型 ws。 * 從預訓練模型中載入 BERT 分詞器 tokenizer 和 BERT 模型 model。   2.定義函數 get\_bert\_embedding   * 輸入參數: text（文本字符串） * 使用 BERT 分詞器將文本轉換為張量格式。 * 通過 BERT 模型計算嵌入（embedding）。 * 返回文本的最後一層隱藏狀態（CLS token 的嵌入）。   3.定義函數 load\_json   * 輸入參數: file\_path（JSON 文件路徑） * 從指定路徑讀取 JSON 文件並返回解析結果。   4.定義函數 preprocess\_data   * 輸入參數: items（JSON 數據項列表） * 對於每個項目 item：   + 提取並拼接地址文本。   + 使用 CKIP 斷詞工具對地址文本進行斷詞。   + 使用 BERT 模型計算地址文本的嵌入。   + 將嵌入結果存儲在項目中。 * 返回處理過的數據 items。   5.定義函數 cosine\_similarity   * 輸入參數: tensor1, tensor2（兩個張量） * 計算並返回兩個張量之間的餘弦相似度。   6.定義函數 device\_similarity   * 輸入參數: devices1, devices2（兩組設備列表） * 計算兩組設備之間的匹配度，並返回是否匹配度超過 70%。   7.定義函數 get\_tokenized\_text   * 輸入參數: text（文本字符串） * 使用 CKIP 斷詞工具對文本進行斷詞並返回斷詞結果。   8.定義函數 compare\_patterns   * 輸入參數: pattern1, pattern2（兩個房屋模式字符串） * 將兩個模式進行斷詞並計算 BERT 嵌入。 * 計算兩個嵌入之間的餘弦相似度並返回是否相似度超過 90%。   9.定義函數 compare\_layers   * 輸入參數: layer1, layer2（兩個房屋樓層字符串） * 將樓層字符串分割為單層，對每對單層進行斷詞並計算 BERT 嵌入。 * 計算每對嵌入之間的餘弦相似度，返回最大相似度是否超過 90%。   10.定義函數 find\_similar\_items   * 輸入參數: data（已處理的完整數據） * 初始化 similar\_items 列表。 * 對於每對數據項目 data[i] 和 data[j]，檢查以下條件：   + 地址相似度是否超過 90%。   + 模式、大小和樓層的相似度是否均超過指定閾值。   + 設備列表是否匹配。   + 如果上述條件均滿足，將 data[i] 和 data[j] 的 hid 添加到 similar\_items 列表。 * 返回 similar\_items。   11.主程式入口 main   * 調用 load\_json 函數以讀取 JSON 數據。 * 調用 preprocess\_data 函數以預處理數據，獲取完整數據 complete\_data。 * 調用 find\_similar\_items 函數以查找類似項目。 * 打印找到的類似項目，最多顯示前三個。   12.主程式執行   * 調用 main 函數。 | | | |

表 9-2‑9虛擬碼- Yolo+clip.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-6-2 | 程式名稱 | Yolo+clip.py |
| 目的 | 房屋圖片的物件偵測與相似度比較。 | | |
| 虛擬碼 | | | |
| 1.初始化與設置   * 載入所需的模組和庫：   + torch 用於深度學習相關操作。   + PIL.Image 用於圖像處理。   + transformers.CLIPProcessor 和 transformers.CLIPModel 用於 CLIP 模型的處理和推理。   + ultralytics.YOLO 用於 YOLO 物件偵測模型。   + collections.Counter 用於計算圖像的主要顏色。   + numpy 用於數據處理。   + sklearn.metrics.pairwise\_distances 用於計算顏色相似度。   + os 和 glob 用於文件系統操作。   2.初始化模型   * 初始化 YOLO 模型 yolo\_model。 * 從預訓練模型中載入 CLIP 模型 clip\_model 和 CLIP 處理器 clip\_processor。   3.定義函數 detect\_objects   * 輸入參數: image\_path（圖像路徑） * 使用 YOLO 模型進行物件偵測。 * 返回偵測到的物件座標和結果。   4.定義函數 get\_dominant\_color   * 輸入參數: image（PIL 圖像對象） * 將圖像調整為 50x50 大小，提取像素並計算主要顏色。 * 返回主要顏色。   5.定義函數 generate\_clip\_description   * 輸入參數: image（PIL 圖像對象），objects（偵測到的物件） * 對每個物件進行以下操作：   + 裁剪圖像中對應的物件部分。   + 計算裁剪圖像的主要顏色。   + 使用 CLIP 模型生成該物件的圖像嵌入。   + 定義一組包含顏色信息的文本描述。   + 使用 CLIP 模型生成文本嵌入。   + 計算圖像特徵和文本特徵之間的相似度，找出最匹配的文本描述。   + 將匹配的描述和顏色信息添加到描述列表中。 * 返回生成的描述列表。   6.定義函數 calculate\_similarity   * 輸入參數: desc1（描述列表1），desc2（描述列表2） * 初始化相似度列表 similarities。 * 對每對描述進行以下操作：   + 計算文本相似度（忽略顏色部分）。   + 計算顏色相似度（使用餘弦相似度）。   + 結合文本和顏色相似度，計算綜合相似度。   + 將綜合相似度添加到相似度列表中。 * 返回相似度列表。   7.定義函數 process\_folder   * 輸入參數: folder\_path（資料夾路徑） * 使用 glob 模組獲取資料夾中的所有圖像文件路徑。 * 初始化描述字典 all\_descriptions。 * 對每個圖像文件進行以下操作：   + 調用 detect\_objects 函數偵測物件。   + 使用 PIL 載入圖像。   + 調用 generate\_clip\_description 函數生成圖像描述。   + 將描述存儲到 all\_descriptions 字典中。   + 打印處理信息。 * 返回所有圖像的描述字典。   8.處理資料夾並比較相似度   * 定義兩個資料夾路徑 folder1 和 folder2。 * 調用 process\_folder 函數處理這兩個資料夾，獲取每個圖像的描述。 * 初始化相似度列表 all\_similarities。 * 對每對圖像描述進行以下操作：   + 如果描述列表不為空，調用 calculate\_similarity 函數計算相似度。   + 計算平均相似度並存儲結果。 * 根據設定的閾值判斷兩個資料夾內圖像的相似程度。 * 打印相似度結果和資料夾相似度比例。   9.主程式入口   * 執行資料夾處理和相似度比較邏輯，並打印最終結果。 | | | |

表 9-2‑10虛擬碼-ner\_clean\_1.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-6-3 | 程式名稱 | ner\_clean\_1.py |
| 目的 | 提取房屋描述內有關附近生活機能的店家或設施。 | | |
| 虛擬碼 | | | |
| 1.初始化與設置   * 定義輸入文件路徑 input\_file\_path 和輸出文件路徑 output\_file\_path。 * 載入 JSON 文件中的數據。   2.定義自訂後綴與排除項   * 定義一組表示機構的自訂後綴 custom\_suffixes（如 "公園", "學校", "醫院" 等）。 * 定義一組需要排除的特定項目 exclusions。   3.定義函數 preprocess\_names   * 輸入參數: names（機構名稱列表） * 定義分割模式 split\_patterns，針對某些特殊情況對名稱進行分割和處理。 * 初始化 processed\_names 列表。 * 對於每個名稱 name：   + 檢查名稱是否符合 split\_patterns 中的任何模式。   + 如果符合，將其分割為多個名稱並添加到 processed\_names 列表中。   + 否則，將原始名稱添加到 processed\_names 列表中。 * 返回處理過的名稱列表 processed\_names。   4.定義函數 remove\_duplicates\_and\_manage\_empty   * 輸入參數: data（處理過的 JSON 數據） * 對於每個數據項 item：   + 將 store 列表轉換為集合，以移除重複項目。   + 如果集合中存在“醫院”，則確保集合中不會有“醫療機構”。   + 如果集合中沒有“醫院”，但存在“醫療機構”，則將“醫療機構”替換為“醫院”。   + 如果集合中存在“學校”，並且集合中包含特定的教育後綴（如“高中”、“小學”、“幼兒園”、“大學”、“學院”），則移除“學校”。   + 將集合轉換回列表，或者如果集合為空，則將 store 設置為 None。 * 返回處理過的數據 data。   5.保存處理後的數據   * 將處理過的數據保存到指定的輸出文件路徑 output\_file\_path 中，格式為 JSON。   6.打印完成訊息   * 打印消息，表明處理過的數據已保存到輸出文件路徑。 | | | |

# 

## 測試計畫

本系統執行所需的硬體為作業系統Android 8.0以上的智慧型手機。在系統程式開發設計完成後，將程式存為APK檔後放在手機上測試，確認每項功能是否皆能正確執行。

1. 會員註冊：確認使用者能夠註冊。
2. 會員登入：確認使用者能夠登入到首頁。
3. 忘記密碼：使用者忘記密碼，輸入電子郵件後，能夠成功接收到驗證碼。
4. 更改會員資料：
5. 更改密碼：修改使用者密碼。
6. 地區搜尋：
7. 捷運搜尋：
8. 刊登房屋：
9. 幫刊登者自動算出離捷運公車距離：
10. 編輯刊登的房屋資訊：
11. 訂閱新增及查看：
12. 收藏房屋：
13. 房屋瀏覽紀錄：
14. 房屋詳細資訊查看：
15. Line 智能助手：

## 測試個案與測試結果資料

|  |  |
| --- | --- |
| 功能名稱 | 會員註冊 |
| 測試目標 | 確認使用者能夠註冊 |
| 測試作業 | 1. 輸入信箱與密碼。 2. 驗證資料庫是否有重複的資訊。 |
| 測試結果 | 成功 |

|  |  |
| --- | --- |
| 功能名稱 | 會員登入 |
| 測試目標 | 確認使用者能夠登入到首頁 |
| 測試作業 | 1. 輸入信箱與密碼。 2. 驗證資料庫是否有相同的資訊。 |
| 測試結果 | 成功 |

|  |  |
| --- | --- |
| 功能名稱 | 忘記密碼 |
| 測試目標 | 使用者忘記密碼，輸入電子郵件後，能夠成功接收到驗證碼。 |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 更改會員資料 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 更改密碼 |
| 測試目標 | 修改使用者密碼 |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 地區搜尋 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 捷運搜尋 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 刊登房屋 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 幫刊登者自動算出離捷運公車距離 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 編輯刊登的房屋資訊 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 訂閱新增及查看 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 收藏房屋 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 房屋瀏覽紀錄 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | 房屋詳細資訊查看 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

|  |  |
| --- | --- |
| 功能名稱 | Line 智能助手 |
| 測試目標 |  |
| 測試作業 |  |
| 測試結果 |  |

# 操作手冊

# 使用手冊

# 感想

# 參考資料

|  |  |
| --- | --- |
| **[1]** 政府資料開放平台-大專院校校別學生數：<https://data.gov.tw/dataset/6231> | |
| **[2]** 網站流量分析statcounter：[https://gs.statcounter.com/os-market-share](https://gs.statcounter.com/os-market-share%20) | |
| **[3]** iPhone 和 Android 收入和用戶統計 (2024)：<https://makemoney.onlinemarketingleader.com/2024/03/14/iphone-%E5%92%8C-android-%E6%94%B6%E5%85%A5%E5%92%8C%E7%94%A8%E6%88%B6%E7%B5%B1%E8%A8%88-2024/> | |
| **[4]** 中研院 CKIP Transformers 中文斷詞：<https://ithelp.ithome.com.tw/articles/10295882> | |
| **[5]** 進擊的BERT：NLP界的巨人之力與遷移學習：<https://leemeng.tw/attack_on_bert_transfer_learning_in_nlp.html> | |
| **[6]** 爬蟲-Selenium：<https://www.selenium.dev/documentation/> | |
| **[7]** dart和flutter的第三方套件平台：<https://pub.dev/> | |
| **[8]** flutter的icon表：<https://www.fluttericon.cn/> | |
| **[9]**展示 Flutter 框架多功能性和多樣化 UI 元件的平台：  <https://flutter-gallery-archive.web.app/> | |
| **[10]**《Keras大神歸位：深度學習全面進化！用 Python 實作CNN、RNN、GRU、LSTM、GAN、VAE、Transformer》François Chollet (2022)-旗標科技 | |
| **[11]**【資料庫】SQL 3小時初學者教學：<https://youtu.be/gvRXjsrpCHw?si=DGJVvTLlph4OTKb2> | |
| **[12]** MySQL Workbench實體關係模型介面操作簡介：  <https://youtu.be/ygP-W8iKElw?si=ja8vdmZ8wysL1UGg> |
| **[13]**GitHub-ckiplab/ckiptagger：<https://github.com/ckiplab/ckiptagger> | |

|  |
| --- |
| **[14]**GitHub-google-research/bert：  <https://github.com/google-research/bert> |
| **[15]** GitHub-openai/CLIP：<https://github.com/openai/CLIP> |
| **[16]** GitHub-ultralytics/ultralytics(YOLOv8)：<https://github.com/ultralytics/ultralytics> |

# 附錄一 會議記錄

本組每周都會與老師開會討論，不過因文件篇幅有限，所以只擷取幾次較為重要的會議紀錄：













# 附錄二 問卷題目

在設定問卷調查題目時，我們根據填答者是否有租屋經驗來分配不同的問題區塊。以下列出的問題，是我們在這次調查中較為主要的題目：















# 附錄三 評審問題回覆

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 591的資料取得，不利未來軟體上線，會有授權問題，591沒有授權爬蟲，應該針對多元的資料來源 | 爬蟲資料只是初始資料來源，用來支持初始系統運作而已，不需要591的資料也可以運作，所以也就不需要從更多的平台爬蟲取得房源資料 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 試著利用AI在租屋上找出更大的用途,僅用在字串或圖片比對有點可惜  (或許有演算法可替代) | 主要是將重複的內容提取出來，將**兩句意思一樣的話透過AI比對**，不僅僅只用來比對字串或是圖片  (例:樓下有7-11、7-11在樓下) |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 主題是環繞在ai的部分嗎?為何強調重複物件刊登? | 在使用其他app的時候發現很多重複的房屋物件，造成找房時間的浪費，所以我們致力於將這個最大的缺點改善 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| Bert的細節?Bert是比對什麼東西，針對地址的部分要再思考一下 | Bert針對房屋的所有文字類資訊做對比 使用Bert才可對不同文字同個意思的文字資訊做篩選 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 為何要有Line bot，不直接在App裡面寫一個聊天室就好 |  |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 租屋平台的重點，應該要是推薦(AI)房源、可以適合去比賽或是引進ShopBack利用導流可以產學合作(拆帳分潤導流) |  |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 房源不足要如何增加使用者使用，如何吸引大眾 |  |