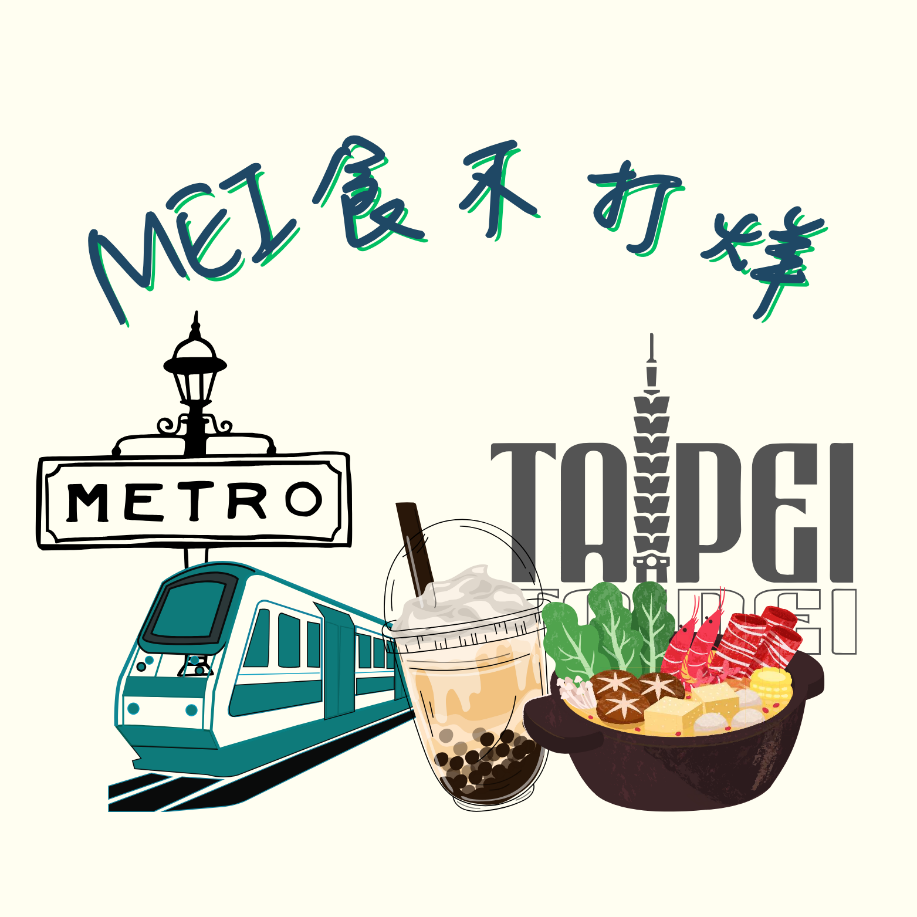
國立臺北商業大學

資 訊 管 理 系

114，資訊系統專案設計

**系統手冊**

****

**組別：第114407組**

**題目：MEI食不打烊**

**指導老師：葉明貴系主任**

**組長：11146061 郭宥妍**

**組員：11146066 簡孝宇 11146073 張楷偉  
11146083 林勝威**

**中華民國114年10月08日**

**目錄**

[第1章　前言 1](#_Toc210725099)

[1-1 背景介紹 1](#_Toc210725100)

[1-2 動機 1](#_Toc210725101)

[1-3 系統目的與目標 2](#_Toc210725102)

[1-4 預期成果 2](#_Toc210725103)

[第2章　營運計畫 3](#_Toc210725104)

[2-1 可行性分析 3](#_Toc210725105)

[2-2 商業模式－Business model 4](#_Toc210725106)

[2-3 市場分析－STP 5](#_Toc210725107)

[2-4 競爭力分析SWOT-TOWS 7](#_Toc210725108)

[第3章　系統規格 9](#_Toc210725109)

[3-1 系統架構 9](#_Toc210725110)

[3-2 系統軟、硬體需求與技術平台 11](#_Toc210725111)

[3-3 使用標準與工具 11](#_Toc210725112)

[第4章　專案時程與組織分工 12](#_Toc210725113)

[4-1 專案時程 12](#_Toc210725114)

[4-2 專案組織與分工 13](#_Toc210725115)

[4-3 GitHub紀錄 15](#_Toc210725116)

[第5章　需求模型 16](#_Toc210725117)

[5-1 使用者需求 16](#_Toc210725118)

[5-2 使用個案圖(Use case diagram) 17](#_Toc210725119)

[5-3 使用個案描述 17](#_Toc210725120)

[5-4 分析類別圖(Analysis class diagram) 20](#_Toc210725121)

[第6章　設計模型 21](#_Toc210725122)

[6-1 循序圖(Sequential diagram) 21](#_Toc210725123)

[6-2 設計類別圖(Design class diagram) 23](#_Toc210725124)

[第7章　實作模型 24](#_Toc210725125)

[7-1 佈署圖(Deployment diagram) 24](#_Toc210725126)

[7-2 套件圖(Package diagram) 24](#_Toc210725127)

[7-3 元件圖(Component diagram) 26](#_Toc210725128)

[7-4 狀態機(State machine) 27](#_Toc210725129)

[第8章　資料庫設計 31](#_Toc210725130)

[8-1 資料庫關聯表 31](#_Toc210725131)

[8-2 表格及其Meta data 32](#_Toc210725132)

[第9章　程式 38](#_Toc210725133)

[9-1 元件清單及其規格描述 38](#_Toc210725134)

[9-2 其他附屬之各種元件 42](#_Toc210725135)

[第10章　測試模型 69](#_Toc210725136)

[10-1 測試計畫 69](#_Toc210725137)

[10-2 測試個案與測試結果 70](#_Toc210725138)

[第11章　操作手冊 72](#_Toc210725139)

[11-1 系統元件 72](#_Toc210725140)

[第12章　使用手冊 73](#_Toc210725141)

[第13章　感想 74](#_Toc210725142)

[第14章　參考資料 76](#_Toc210725143)

[附錄 77](#_Toc210725144)

**圖目錄**

[▲圖 3-1‑1：系統架構圖 9](#_Toc210743609)

[▲圖 3-1‑2：系統流程圖 10](#_Toc210743610)

[▲圖 4-3‑1：上傳GitHub紀錄－11146061 郭宥妍 15](#_Toc210743611)

[▲圖 4-3‑2：上傳GitHub紀錄－11146066 簡孝宇 15](#_Toc210743612)

[▲圖 4-3‑3：上傳GitHub紀錄－11146073 張楷偉 15](#_Toc210743613)

[▲圖 4-3‑4：上傳GitHub紀錄－11146083 林勝威 15](#_Toc210743614)

[▲圖 5-2‑1：使用個案圖 17](#_Toc210743615)

[▲圖 5-4‑1：分析類別圖 20](#_Toc210743616)

[▲圖 6-1‑1：循序圖－登入/註冊 21](#_Toc210743617)

[▲圖 6-1‑2：循序圖－餐廳查詢 21](#_Toc210743618)

[▲圖 6-1‑3：循序圖－餐廳評論 22](#_Toc210743619)

[▲圖 6-1‑4：循序圖－捷運查詢 22](#_Toc210743620)

[▲圖 6-2‑1：設計類別圖 23](#_Toc210743621)

[▲圖 7-1‑1：佈署圖 24](#_Toc210743622)

[▲圖 7-2‑1：套件圖－登入/註冊模組 24](#_Toc210743623)

[▲圖 7-2‑2：套件圖－餐廳管理模組 25](#_Toc210743624)

[▲圖 7-2‑3：套件圖－評論模組 25](#_Toc210743625)

[▲圖 7-2‑4：套件圖－捷運/站點模組 25](#_Toc210743626)

[▲圖 7-2‑5：套件圖－用戶管理模組 26](#_Toc210743627)

[▲圖 7-2‑6：套件圖－首頁/重定向模組 26](#_Toc210743628)

[▲圖 7-3‑1：元件圖 26](#_Toc210743629)

[▲圖 7-4‑1：狀態機－登入/註冊流程 27](#_Toc210743630)

[▲圖 7-4‑2：狀態機－用戶（帳號啟用/停權/管理員） 28](#_Toc210743631)

[▲圖 7-4‑3：狀態機－餐廳營業 28](#_Toc210743632)

[▲圖 7-4‑4：狀態機－評論發佈流程 29](#_Toc210743633)

[▲圖 7-4‑5：狀態機－前端頁面/表單切換 29](#_Toc210743634)

[▲圖 7-4‑6：狀態機－首頁重定向 30](#_Toc210743635)

[▲圖 8-1‑1：MySQL資料庫ER圖 31](#_Toc210743636)

[▲圖 8-1‑2：資料庫關聯圖 32](#_Toc210743637)

[▲圖 11-1‑1：註冊頁面 75](#_Toc210743638)

[▲圖 11-1‑2：登入頁面 75](#_Toc210743639)

**表目錄**

[▼表 2-3‑1：潛在使用者區分為三大族群 5](#_Toc210743549)

[▼表 2-3‑2：平台主打特色 6](#_Toc210743550)

[▼表 2-4‑1：SWOT分析 7](#_Toc210743551)

[▼表 2-4‑2：TOWS分析 8](#_Toc210743552)

[▼表 3-2‑1：系統軟、硬體需求與技術平台 11](#_Toc210743553)

[▼表 3-3‑1：系統環境 11](#_Toc210743554)

[▼表 3-3‑2：系統開發工具 11](#_Toc210743555)

[▼表 3-3‑3：程式開發技術 11](#_Toc210743556)

[▼表 3-3‑4：文件編寫工具 11](#_Toc210743557)

[▼表 3-3‑5：管理程式平台 11](#_Toc210743558)

[▼表 4-1‑1：專案時程時程表 12](#_Toc210743559)

[▼表 4-2‑1：專案組織與分工 13](#_Toc210743560)

[▼表 4-2‑2：工作內容 14](#_Toc210743561)

[▼表 5-1‑1：使用者需求表 16](#_Toc210743562)

[▼表 5-3‑1：登入 17](#_Toc210743563)

[▼表 5-3‑2：註冊 18](#_Toc210743564)

[▼表 5-3‑3：查詢餐廳 18](#_Toc210743565)

[▼表 5-3‑4：查看餐廳詳細資訊 19](#_Toc210743566)

[▼表 5-3‑5：發表評論/打卡 19](#_Toc210743567)

[▼表 8-2‑1：資料表 32](#_Toc210743568)

[▼表 8-2‑2：user 33](#_Toc210743569)

[▼表 8-2‑3：metroline 33](#_Toc210743570)

[▼表 8-2‑4：station 34](#_Toc210743571)

[▼表 8-2‑5：restaurant 34](#_Toc210743572)

[▼表 8-2‑6：businesshours 35](#_Toc210743573)

[▼表 8-2‑7：review 36](#_Toc210743574)

[▼表 8-2‑8：checkinreview 36](#_Toc210743575)

[▼表 8-2‑9：traininfo 37](#_Toc210743576)

[▼表 8-2‑10：stationid 37](#_Toc210743577)

[▼表 9-1‑1：元件清單與規格描述表(前端)－網頁介面撰寫 38](#_Toc210743578)

[▼表 9-1‑2：元件清單與規格描述表(前端)－網頁介面設計 38](#_Toc210743579)

[▼表 9-1‑3：元件清單與規格描述表(前端)－網頁動畫設計 39](#_Toc210743580)

[▼表 9-1‑4：元件清單與規格描述表(後端)－爬蟲 40](#_Toc210743581)

[▼表 9-1‑5：元件清單與規格描述表(後端)－讀取API 40](#_Toc210743582)

[▼表 9-1‑6：元件清單與規格描述表(後端)－Service 41](#_Toc210743583)

[▼表 9-1‑7：元件清單與規格描述表(後端)－定時更新資訊 41](#_Toc210743584)

[▼表 9-1‑8：元件清單與規格描述表(後端)－使用者資訊管理 41](#_Toc210743585)

[▼表 9-1‑9：元件清單與規格描述表(後端)－串接相關網頁 41](#_Toc210743586)

[▼表 9-2‑1：虛擬碼－homepage.html、homepage.css 42](#_Toc210743587)

[▼表 9-2‑2：虛擬碼－Member\_center.html、Member\_center.css 44](#_Toc210743588)

[▼表 9-2‑3：虛擬碼－night.html、night.css 46](#_Toc210743589)

[▼表 9-2‑4：虛擬碼－missiom.js 49](#_Toc210743590)

[▼表 9-2‑5：虛擬碼－stationdata.js 50](#_Toc210743591)

[▼表 9-2‑6：虛擬碼－metro\_train\_service.py 52](#_Toc210743592)

[▼表 9-2‑7：虛擬碼－models.py 55](#_Toc210743593)

[▼表 9-2‑8：虛擬碼－views.py 57](#_Toc210743594)

[▼表 9-2‑9：虛擬碼－serializers.py 62](#_Toc210743595)

[▼表 9-2‑10：虛擬碼－review\_service.py 65](#_Toc210743596)

[▼表 10-2‑1：測試結果－會員註冊 70](#_Toc210743597)

[▼表 10-2‑2：測試結果－會員登入 70](#_Toc210743598)

[▼表 10-2‑3：測試結果－更改會員資料 70](#_Toc210743599)

[▼表 10-2‑4：測試結果－站點查詢 71](#_Toc210743600)

[▼表 10-2‑5：測試結果－當月TOP 10店家 71](#_Toc210743601)

[▼表 10-2‑6：測試結果－夜貓子專區 71](#_Toc210743602)

[▼表 10-2‑7：測試結果－喝酒專區 72](#_Toc210743603)

[▼表 10-2‑8：測試結果－打卡任務評論 72](#_Toc210743604)

[▼表 10-2‑9：測試結果－美食任務版 72](#_Toc210743605)

[▼表 10-2‑10：測試結果－捷運時刻表 73](#_Toc210743606)

[▼表 10-2‑11：測試結果－美食地圖 73](#_Toc210743607)

[▼表 10-2‑12：測試結果－收藏及通知 73](#_Toc210743608)

# 第1章　前言

## 背景介紹

台北捷運系統作為都市運輸的重要骨幹，每日承載大量通勤與觀光人潮，其沿線站點涵蓋多個商圈與飲食聚落，具有高度的區域特色與美食潛力。而現有的美食資訊平台多數以行政區域或餐廳類型分類為主，缺乏以「捷運站」為導向的查詢邏輯，對依賴大眾運輸移動的使用者而言，難以快速掌握特定站點周邊的餐飲資訊。

其次，隨著社群媒體風潮興起，越來越多使用者習慣透過「打卡」、「拍照上傳」等方式分享美食與生活體驗，社群互動逐漸成為資訊擴散的重要來源。導入會員制度與互動任務機制，已成為提升平台活躍度與使用者參與的主流設計方向。

此外，夜間活動人口對於深夜餐飲選擇與交通回程規劃仍面臨資訊不足的問題，顯示夜間生活相關資訊整合的需求仍有成長空間。

## 動機

隨著智慧型手機與網路服務的快速發展，民眾在外出時越來越依賴行動裝置來查詢即時資訊，特別是在交通與美食領域的需求尤為顯著。台北捷運系統作為首都地區最重要的公共運輸網絡，每日串聯數百萬人次的通勤與觀光流動，其各站周邊聚集了大量在地美食資源。然而，目前市面上仍缺乏一個以「捷運站」為核心，整合交通、美食、社群服務的互動平台。

## 系統目的與目標

本系統旨在打造一個以台北捷運站為主軸的美食資訊平台，使用者可透過手機查詢各站周邊的推薦餐廳與深夜美食資訊。平台採用捷運站導向的查詢方式，提升資訊搜尋效率，並結合會員登入、拍照打卡、評論與任務等互動功能，強化社群參與與內容豐富度。此外，設置「夜貓子專區」，整合深夜營業店家與捷運末五班車時刻，協助夜間用戶安排行程。系統也導入推播功能，主動提醒捷運列車即將到站的時間，提升使用者的行動便利性。整體平台支援繁體中文語言，並採用響應式設計，確保行動裝置上的良好操作體驗，打造一個實用、友善且具互動性的美食導覽平台。

## 預期成果

我們預期系統可以以獨家設置「夜貓子專區」，整合深夜營業店家與末班車時刻，專為夜間行動者量身打造，貼心守護每一個夜晚的旅程。平台支援繁體中文與響應式設計，無論身處何地都能輕鬆操作，串聯每位饕客與捷運沿線美味。讓行動導覽與社群美食兼具，成就智慧城市的新型態餐飲探索體驗，進而達到「美味同行，捷運每一站都精彩！」。

# 第2章　營運計畫

## 可行性分析

* 技術可行性

1. 本系統採用成熟穩定的Web技術，包括HTML5、CSS、JavaScript、Fetch API資料處理與響應式設計，並搭配Django等後端框架進行資料串接與展示。
2. 使用者功能包含捷運站導向的美食查詢、夜貓子餐廳推薦、打卡評論與多語操作，皆為可實作功能，已分工執行中。

* 資料可行性

1. 我們實際向台北捷運官方申請了站點資料API，取得各站相關資訊與即時列車資訊作為查詢基礎。
2. 同時也向Google Places API正式申請金鑰，用以補足店家圖片、營業資訊、使用者評論等資料，確保內容即時且具參考價值。
3. 除了API資料，也可搭配必要的手動輸入測試資料與模擬使用者評論來建立測試案例與展示效果。

* 操作可行性

1. 系統導向捷運站點搜尋邏輯，搭配夜間資訊與推播提醒，符合真實情境需求。
2. 支援繁體中文語言，提供觀光友善環境，並優化手機使用體驗，易於推廣與實際使用。

* 實施可行性

1. 已完成API串接測試與資料顯示範例，分工包含前端介面設計、後端串接邏輯、資料處理與互動功能開發，具備落地實作能力。
2. 系統開發過程將持續依照專題進度規劃逐步整合並測試。

## 商業模式－Business model

* 關鍵夥伴（Key Partners）

1. 台北捷運公司（提供站點資訊API）
2. Google Places API（提供餐廳詳細資訊與圖片）
3. 開放資料平台（補足區域與地點資訊）

* 關鍵活動（Key Activities）

1. 建立捷運站導向的美食搜尋系統
2. 串接捷運與Google API資料
3. 設計互動功能（打卡、評論、任務）
4. 提供夜間美食與交通資訊整合
5. 響應式網站開發

* 價值主張（Value Proposition）

1. 台北第一個「以捷運站為中心」的美食與夜貓資訊平台
2. 解決夜間餐飲選擇難
3. 支援繁體中文與行動操作，觀光友善、通勤方便
4. 提供「打卡任務」、「夜間專區」、等創新互動功能

## 市場分析－STP

* 市場區隔Segmentation

依據使用者的行為特性與使用情境，可將潛在使用者區分為下列三大族群：

▼表 2-3‑1：潛在使用者區分為三大族群

| 區隔族群 | 特徵說明 |
| --- | --- |
| 通勤族群 | 以捷運作為每日上下班交通工具，對「午餐、晚餐」推薦有高度需求。使用時間集中於早上與傍晚。 |
| 夜間活動族 | 活動時間為夜間，常有深夜餐飲需求（如學生、夜班族、夜生活族群）。關心餐廳營業時間與回家交通。 |
| 國內自由行旅客 | 在台北捷運沿線移動，對地點不熟，習慣使用手機查詢。常有語言障礙，關心推薦、地圖、評論支援。 |

* 目標對象Targeting

平台的首波目標使用者為：

1. 以台北捷運為主要移動方式的通勤者與觀光客

* 因平台以「捷運站」為搜尋核心，特別適合此類使用者快速找到附近美食。

1. 對夜間生活有需求的年輕族群或上夜班者

* 夜間外食資訊缺乏，平台設置「夜貓子專區」與「末班車時刻」剛好填補空白。
* 產品定位Positioning

「全台第一個以捷運站為導向的多語美食地圖，深夜行動族群的最佳夥伴」

▼表 2-3‑2：平台主打特色

| 定位主軸 | 說明 |
| --- | --- |
| 捷運站導向 | 使用者以車站為起點查找周邊美食，符合實際移動邏輯。 |
| 夜貓子專區 | 整合深夜營業餐廳與末班車資訊，貼近夜間需求。 |
| 社群互動 | 提供拍照打卡、評論任務，提升參與感與內容豐富度。 |

## 競爭力分析SWOT-TOWS

▼表 2-4‑1：SWOT分析

|  |  |
| --- | --- |
| 優勢Strengths | 劣勢Weaknesses |
| * 以捷運站為導向，查詢邏輯清晰直覺 * 結合「夜間營業」與「班車提醒」，功能創新 * 提供打卡、評論、任務互動功能，增加使用體驗 * 系統介面採響應式設計，適用各種裝置操作 | * 初期資料仍須人工補充，規模有限 * 評論與使用者互動資料需時間累積 * 需仰賴外部API（捷運與Google）穩定度 * 系統需持續維護與更新，開發期工作量大 |
| 機會Opportunities | 威脅Threats |
| * 台北捷運系統龐大，站點周邊美食豐富 * 使用者對於夜間資訊與在地推薦有實際需求 * 行動裝置使用普及，適合推動即時查詢功能 * 開放資料與公共API提供資料串接支援 | * 類似平台（如Google Maps、美食社群App）功能重疊 * 多語內容需人工處理，語言品質需控管 * API數據可能改版或停止提供，存在依賴風險 * 資料更新頻率與正確性會影響使用者信任 * 多語內容維護成本高，翻譯準確度需控制 |

▼表 2-4‑2：TOWS分析

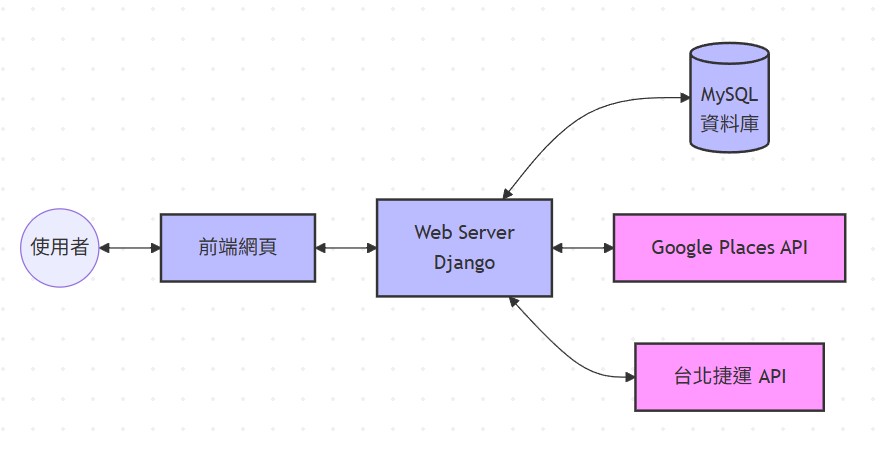
|  |  |  |
| --- | --- | --- |
| 內部因素  外在因素 | 優勢Strengths | 劣勢Weaknesses |
| 機會Opportunities | SO策略（增長性策略）   * 利用捷運導向與夜間資訊整合特色，解決使用者移動與用餐交會問題 * 運用中文介面設計，提升對國內觀光族的實用性與友善度 * 採模組化架構，未來可擴展至其他縣市捷運系統 | WO策略（移轉性策略）   * 初期聚焦重點站點，逐步擴充內容並建立測試資料 * 結合API與手動輸入方式補足資料來源，提高資料覆蓋率 * 強化錯誤回報與使用回饋機制，逐步優化功能 |
| 威脅Threats | ST策略（多角化策略）   * 強調系統特色（捷運導向＋班車提醒＋夜間美食）與現有平台做出區隔 * 建立API備援處理流程，減少依賴性造成的風險 * 設計簡潔、操作直觀，提升易用性以留住使用者 | WT策略（防禦型策略）   * 提供基礎版本測試範圍，確保資料正確與功能完整後再逐步擴展 * 強化多語翻譯管理，方便維護 * 規劃資料更新流程與開發紀錄，方便未來交接或維護 |

綜合SWOT與TOWS分析可見，本系統具備以「捷運站」為導向的查詢優勢，搭配夜間美食資訊、末班車時刻支援，成功區隔於現有以地區或餐廳類型為主的美食平台，展現明確的功能特色與實用價值。

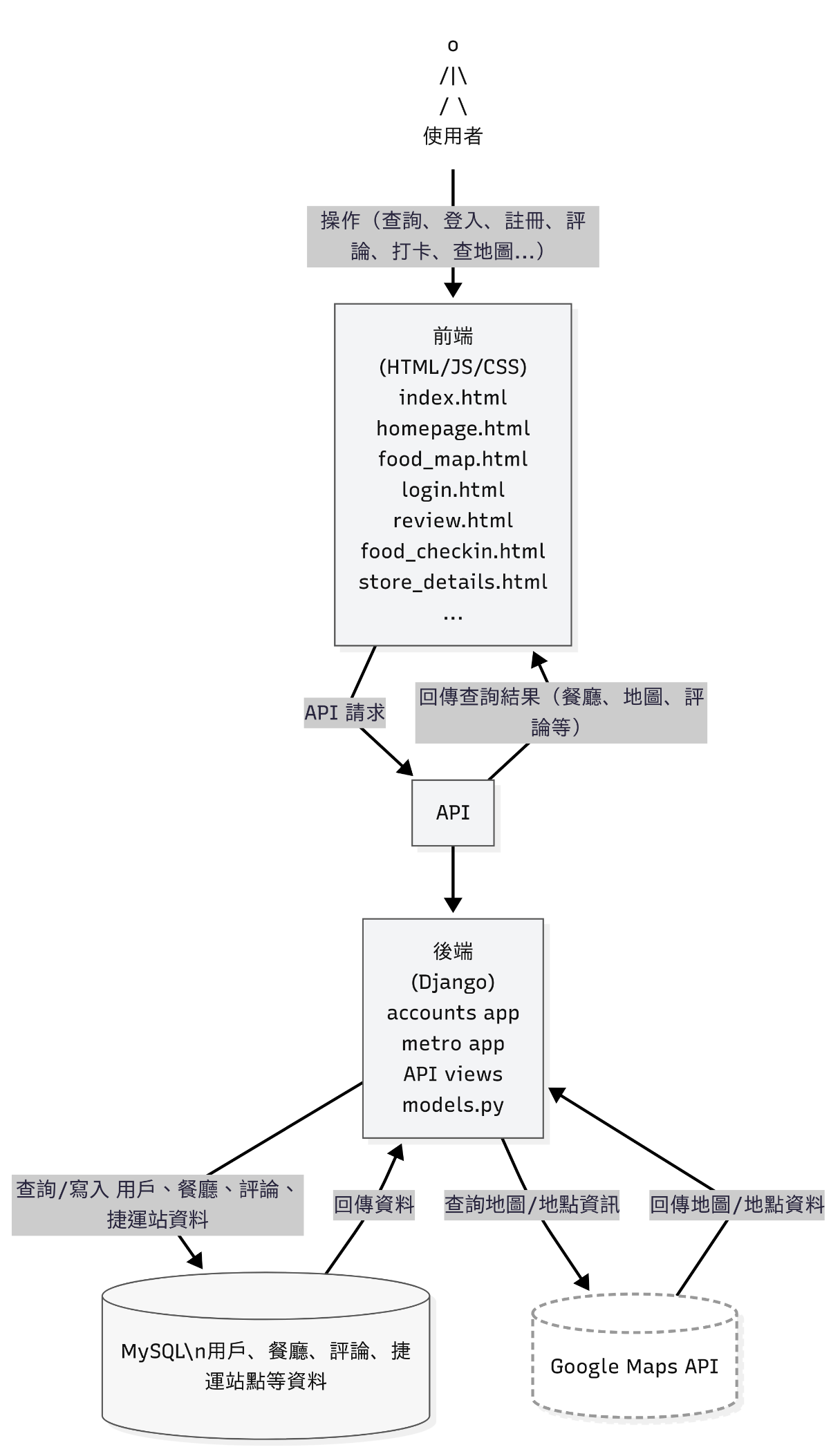
然而，系統初期仍面臨資料覆蓋率不足、API依賴度高與語言維護成本等挑戰，須透過階段性擴充、錯誤回報機制與多語翻譯策略逐步克服。透過模組化設計與漸進式開發，可有效提升平台穩定性與可維護性，並為未來推展至其他縣市或新增功能保留彈性。

# 第3章　系統規格

## 系統架構



▲圖 3-1‑1：系統架構圖



▲圖 3-1‑2：系統流程圖

## 系統軟、硬體需求與技術平台

▼表 3-2‑1：系統軟、硬體需求與技術平台

|  |  |
| --- | --- |
| 設備 | 規格 |
| 手機裝置 | |
| 作業系統 | iOS、Android |
| 版本 | iOS全系列、Android 8以上 |
| 操作介面 | Safari、Chrome |

## 使用標準與工具

▼表 3-3‑1：系統環境

|  |  |
| --- | --- |
| 作業系統 | Windows 11 |
| 網路需求 | WiFi/有線網路 |

▼表 3-3‑2：系統開發工具

|  |  |
| --- | --- |
| 網頁設計 | Visual Studio Code |
| 資料庫 | MySQL |
| 伺服器工具 | Django |

▼表 3-3‑3：程式開發技術

|  |  |
| --- | --- |
| 前端技術 | HTML、CSS、JavaScript |
| 後端技術 | Python、Django |
| API串接 | RESTful API、Google Places API、台北捷運API |

▼表 3-3‑4：文件編寫工具

|  |  |
| --- | --- |
| 繪圖工具 | Canva、Flaticon |
| 簡報製作 | Microsoft PowerPoint 2016 |
| 文件製作 | Microsoft Word 2021 |

▼表 3-3‑5：管理程式平台

|  |  |
| --- | --- |
| 專案管理 | github、fork |

# 第4章　專案時程與組織分工

## 專案時程

▼表 4-1‑1：專案時程時程表

預期進度　實際進度

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1月 | | | 2月 | | | 3月 | | | 4月 | | | 5月 | | | 6月 | | | 7月 | | | 8月 | | | 9月 | | | 10月 | | | 11月 | | | 12月 | | |
| 主題構思 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 相關資料蒐集 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 系統功能分析 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 系統模型 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| UI/UX設計 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Logo設計 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Web開發 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 資料庫設計 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 資料庫建立 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| API設定 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 後端開發 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 系統測試 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 系統整合 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 初評系統手冊 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 複評系統手冊 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 系統簡報 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

## 專案組織與分工

▼表 4-2‑1：專案組織與分工

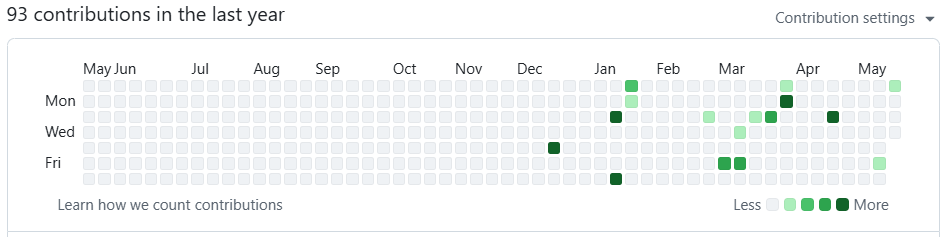
●主要負責人 ○次要負責人

| 組　員  項　目 | | 11146061 郭宥妍 | 11146066 簡孝宇 | 11146073 張楷偉 | 11146083 林勝威 |
| --- | --- | --- | --- | --- | --- |
| 後端開發 | 資料庫建置 |  |  |  | ● |
| 伺服器架設 |  |  |  | ● |
| API設定 |  |  |  | ● |
| 捷運資料轉換 |  |  |  | ● |
| 前端開發 | 登入/註冊 | ○ |  | ○ | ● |
| 首頁 | ○ | ○ | ● |  |
| 夜貓子專區 | ○ | ○ | ● |  |
| 美食打卡 | ○ | ○ | ● |  |
| 捷運時刻表 |  | ○ | ● |  |
| 美食地圖 |  | ○ | ● |  |
| 通知 |  | ○ | ● |  |
| 會員中心 |  | ○ | ● |  |
| Javascript程式 |  | ● |  |  |
| 美術設計 | UI / UX | ● |  |  |  |
| Web介面設計 | ● |  | ○ |  |
| 色彩設計 | ● |  |  |  |
| Logo設計 | ● |  |  |  |
| 素材設計 | ● |  |  |  |
| 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 | ● |  |  |  |
| 報告 | 簡報製作 |  | ● |  |  |

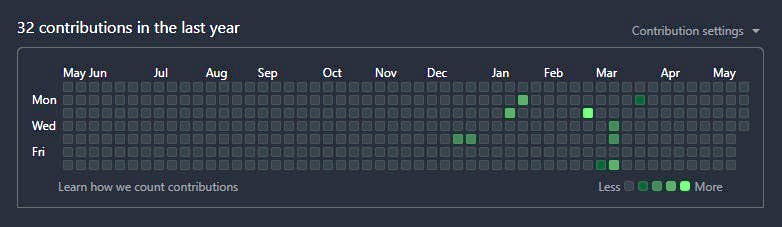
▼表 4-2‑2：工作內容

|  |  |  |  |
| --- | --- | --- | --- |
| 序號 | 姓名 | 工作內容 | 貢獻度 |
| 1 | 組長  郭宥妍 | 文件撰寫、文件排版、文件統整、Logo設計、UI/UX設計 | 25% |
| 2 | 組員  張楷偉 | 前端網頁設計、前端開發 | 20% |
| 3 | 組員  簡孝宇 | 前端動畫開發、串接API、簡報製作、文件撰寫 | 25% |
| 4 | 組員  林勝威 | 前後端串接、後端開發、資料庫建置、伺服器架設、API設定 | 30% |
|  | | | 總計：100% |

## GitHub紀錄



▲圖 4-3‑1：上傳GitHub紀錄－11146061 郭宥妍

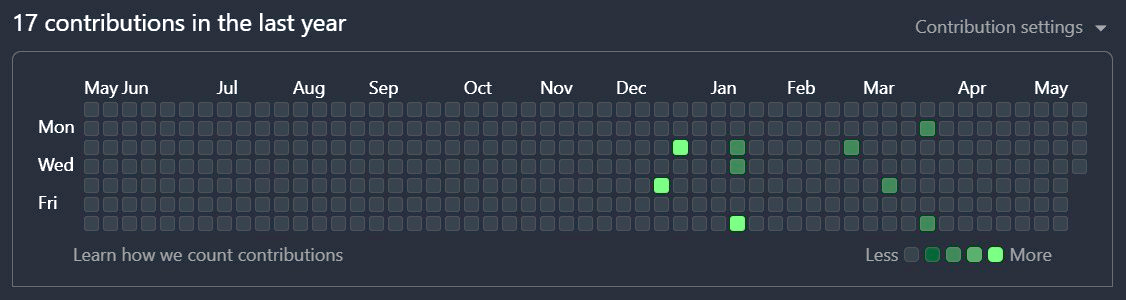


▲圖 4-3‑2：上傳GitHub紀錄－11146066 簡孝宇

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

▲圖 4-3‑3：上傳GitHub紀錄－11146073 張楷偉



▲圖 4-3‑4：上傳GitHub紀錄－11146083 林勝威

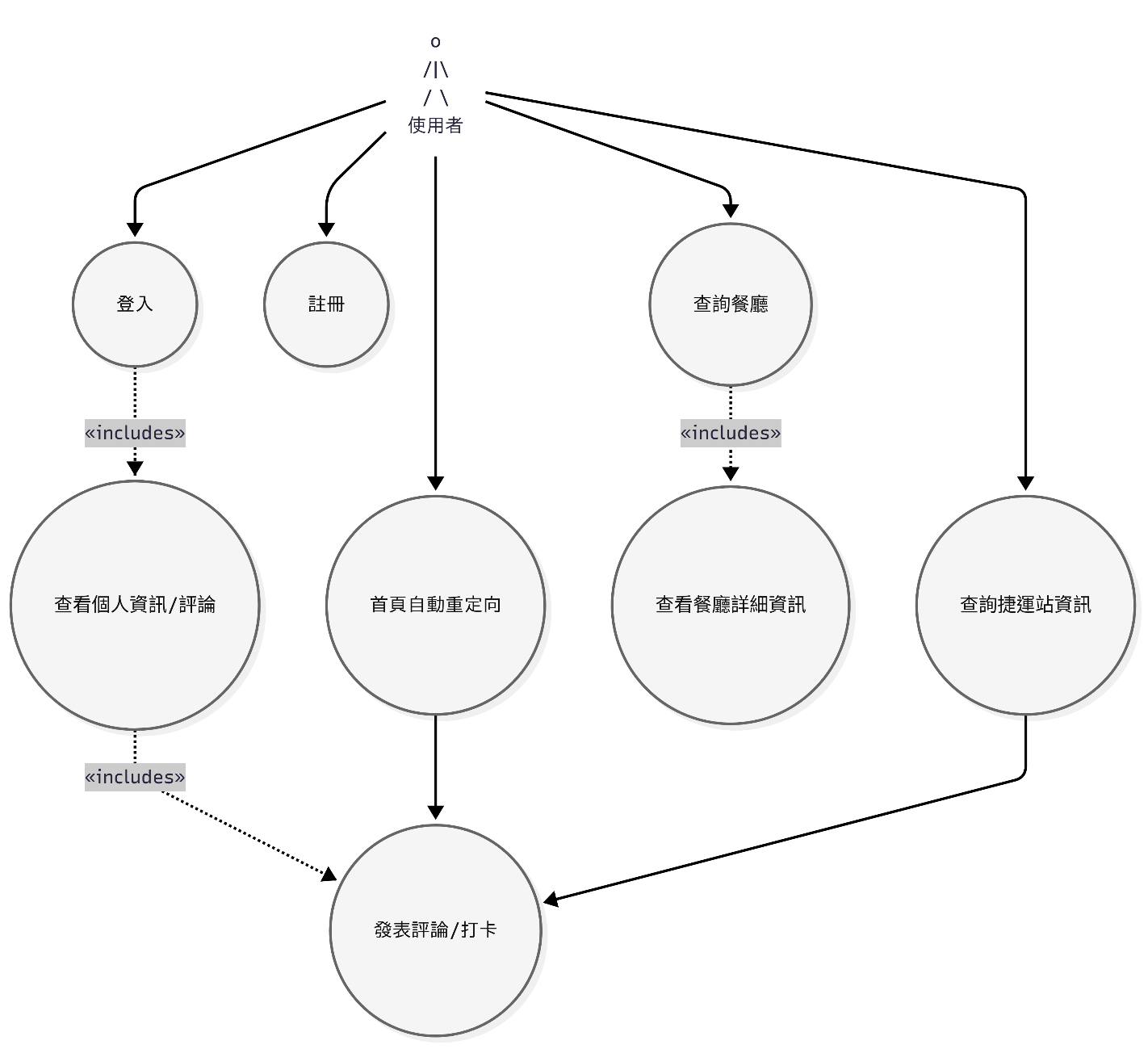
# 第5章　需求模型

## 使用者需求

▼表 5-1‑1：使用者需求表

|  |  |
| --- | --- |
| UseCase | 需求說明 |
| 登入與註冊 | 使用者能夠透過電子郵件註冊帳號並登入系統，進行個人化操作。 |
| 餐廳/捷運站查詢 | 使用者可以透過捷運站、餐廳名稱或地圖查詢附近的餐廳，快速找到想要的美食。 |
| 查看餐廳詳細資訊 | 使用者能夠查看餐廳的詳細資料（名稱、地址、評分、分類、營業時間、評論等）。 |
| 發表評論/打卡 | 使用者可以針對餐廳發表評論、評分，或進行美食打卡，分享用餐體驗。 |
| 查詢捷運站資訊 | 使用者可以查詢捷運站的基本資訊、所屬路線及附近的餐廳。 |
| 查看個人資訊/評論 | 使用者可以在個人頁面查看自己的基本資料、發表過的評論紀錄。 |
| 首頁自動重定向 | 使用者進入首頁時，系統會自動判斷登入狀態並導向對應頁面（已登入導向主頁，未登入導向登入頁）。 |

## 使用個案圖(Use case diagram)



▲圖 5-2‑1：使用個案圖

## 使用個案描述

▼表 5-3‑1：登入

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：登入 | | |
| 主要參與者 | 使用者 | |
| 前置條件 | 使用者已註冊帳號 | |
| 事件路徑： | | |
| 流程 | | 系統回應 |
| 1.使用者輸入信箱與密碼 | |  |
| 2.按下登入 | |  |
| 3.系統驗證帳密 | | 顯示登入成功或失敗訊息 |
| 4.登入成功導向首頁 | |  |

▼表 5-3‑2：註冊

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：註冊 | | |
| 主要參與者 | 使用者 | |
| 前置條件 | 無（新用戶） | |
| 事件路徑： | | |
| 流程 | | 系統回應 |
| 1.使用者輸入信箱、密碼、確認密碼 | |  |
| 2.按下註冊 | |  |
| 3.系統驗證資料 | | 顯示註冊成功或失敗訊息 |
| 4.註冊成功導向登入頁 | |  |

▼表 5-3‑3：查詢餐廳

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：查詢餐廳 | | |
| 主要參與者 | 使用者 | |
| 前置條件 | 無 | |
| 事件路徑： | | |
| 流程 | | 系統回應 |
| 1.用者於首頁、地圖頁、夜市頁等選擇捷運站或輸入關鍵字 | |  |
| 2.系統顯示符合條件的餐廳列表或地圖標記 | |  |
| 3.點擊餐廳可查看詳細資訊 | | 顯示餐廳列表、地圖標記與詳細資訊 |

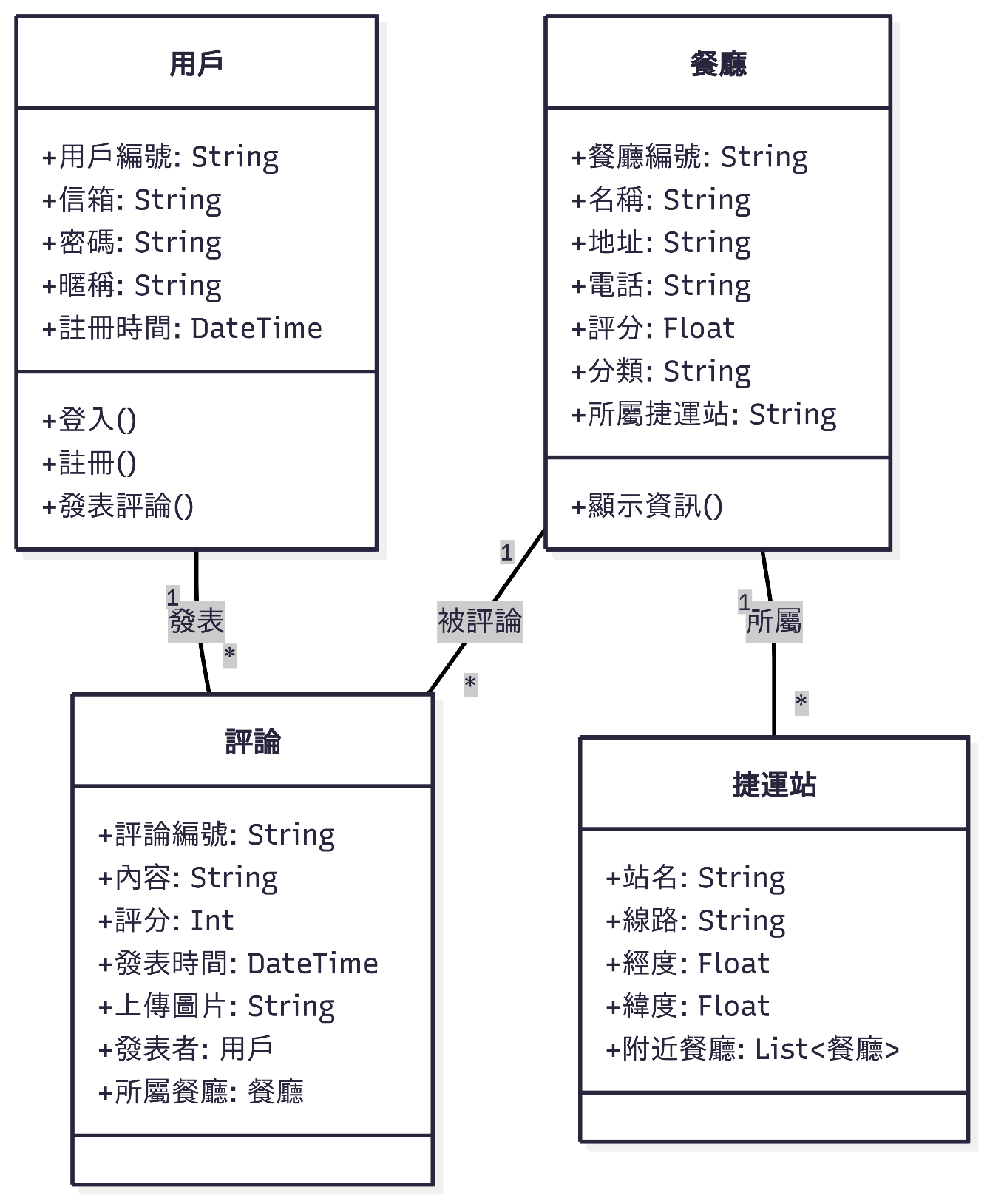
▼表 5-3‑4：查看餐廳詳細資訊

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：查看餐廳詳細資訊 | | |
| 主要參與者 | 使用者 | |
| 前置條件 | 無 | |
| 事件路徑： | | |
| 流程 | | 系統回應 |
| 1.使用者於餐廳列表或地圖點擊餐廳 | |  |
| 2.系統顯示餐廳詳細頁 | | 顯示餐廳詳細資訊 |

▼表 5-3‑5：發表評論/打卡

|  |  |  |
| --- | --- | --- |
| 使用個案名稱：發表評論/打卡 | | |
| 主要參與者 | 使用者 | |
| 前置條件 | 使用者已登入 | |
| 事件路徑： | | |
| 流程 | | 系統回應 |
| 1.進入餐廳頁面或打卡頁 | |  |
| 2.輸入評論內容、評分 | |  |
| 3.按下送出 | | 顯示評論/打卡成功或失敗訊息 |
| 4.系統儲存評論/打卡資料 | |  |

## 分析類別圖(Analysis class diagram)



▲圖 5-4‑1：分析類別圖

# 第6章　設計模型

## 循序圖(Sequential diagram)

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

AI 產生的內容可能不正確。

▲圖 6-1‑1：循序圖－登入/註冊

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

AI 產生的內容可能不正確。

▲圖 6-1‑2：循序圖－餐廳查詢

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

AI 產生的內容可能不正確。

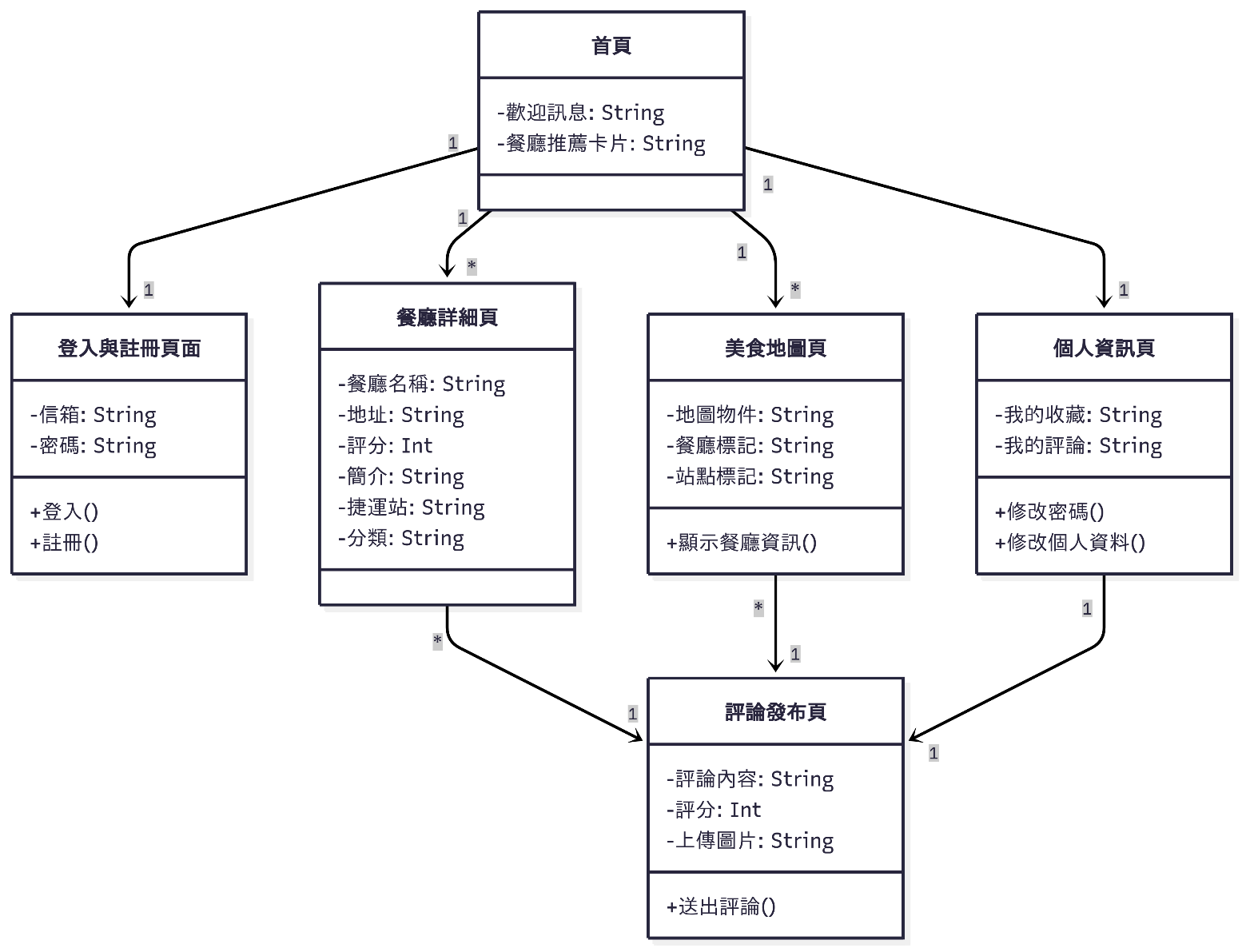
▲圖 6-1‑3：循序圖－餐廳評論

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

AI 產生的內容可能不正確。

▲圖 6-1‑4：循序圖－捷運查詢

## 設計類別圖(Design class diagram)



▲圖 6-2‑1：設計類別圖

# 第7章　實作模型

## 佈署圖(Deployment diagram)

一張含有 文字, 圖表, 螢幕擷取畫面, 方案 的圖片

AI 產生的內容可能不正確。

▲圖 7-1‑1：佈署圖

## 套件圖(Package diagram)

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

▲圖 7-2‑1：套件圖－登入/註冊模組

一張含有 文字, 螢幕擷取畫面, 字型, Rectangle 的圖片

AI 產生的內容可能不正確。

▲圖 7-2‑2：套件圖－餐廳管理模組

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

▲圖 7-2‑3：套件圖－評論模組

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

▲圖 7-2‑4：套件圖－捷運/站點模組

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

AI 產生的內容可能不正確。

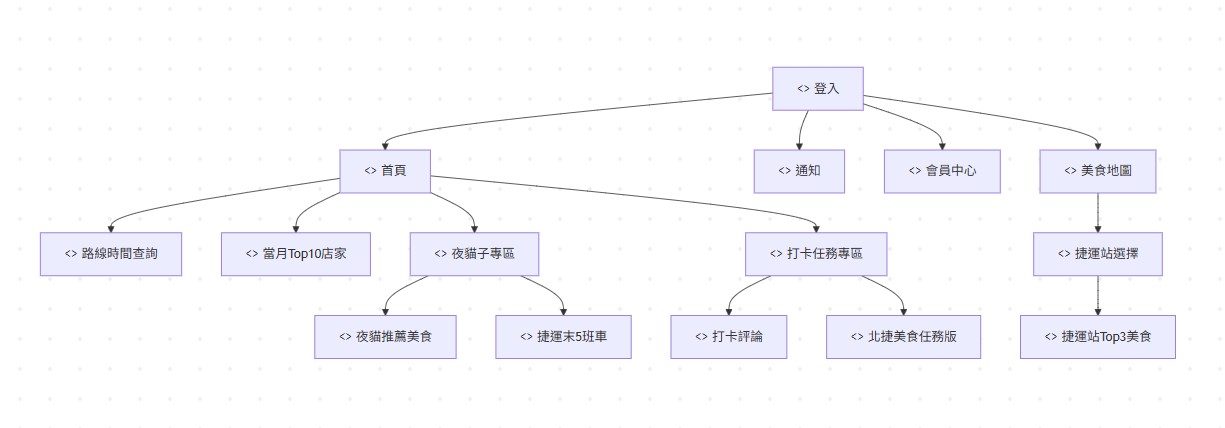
▲圖 7-2‑5：套件圖－用戶管理模組

一張含有 文字, 螢幕擷取畫面, 數字, 字型 的圖片

AI 產生的內容可能不正確。

▲圖 7-2‑6：套件圖－首頁/重定向模組

## 元件圖(Component diagram)



▲圖 7-3‑1：元件圖

## 狀態機(State machine)

一張含有 圖表, 文字, 方案, 寫生 的圖片

AI 產生的內容可能不正確。

▲圖 7-4‑1：狀態機－登入/註冊流程

一張含有 圖表, 文字, 圖畫, 行 的圖片

AI 產生的內容可能不正確。

▲圖 7-4‑2：狀態機－用戶（帳號啟用/停權/管理員）

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

AI 產生的內容可能不正確。

▲圖 7-4‑3：狀態機－餐廳營業

一張含有 文字, 圖表, 寫生, 圖畫 的圖片

AI 產生的內容可能不正確。

▲圖 7-4‑4：狀態機－評論發佈流程

**一張含有 文字, 字型, 圖表, 設計 的圖片

AI 產生的內容可能不正確。**

▲圖 7-4‑5：狀態機－前端頁面/表單切換

**一張含有 文字, 字型, 螢幕擷取畫面, 圖表 的圖片

AI 產生的內容可能不正確。**

▲圖 7-4‑6：狀態機－首頁重定向

# 第8章　資料庫設計

## 資料庫關聯表



▲圖 8-1‑1：MySQL資料庫ER圖

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

AI 產生的內容可能不正確。

▲圖 8-1‑2：資料庫關聯圖

## 表格及其Meta data

▼表 8-2‑1：資料表

|  |  |  |
| --- | --- | --- |
| 編號 | 資料表英文名稱 | 中文名稱/用途 |
|  | user | 使用者資料 |
|  | metroline | 捷運線路 |
|  | station | 捷運站 |
|  | restaurant | 餐廳 |
|  | businesshours | 餐廳營業時間 |
|  | review | 餐廳評論 |
|  | checkinreview | 打卡評論 |
|  | traininfo | 列車到站資訊 |
|  | stationid | 車站ID對照表 |

1. user（使用者資料）

▼表 8-2‑2：user

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 使用者資料 | | 資料表編號 | | 01 |
| 英文名稱 | user | | 主索引 | | id |
| 資料檔陳述 | 紀錄每一位使用者的個人資料及帳號、密碼 | | | | |
| 欄位名稱 | 型態 | 長度/限制 | | 說明 | |
| id | AutoField |  | | 主鍵，自動遞增 | |
| password | CharField | 128 | | 密碼（加密儲存） | |
| last\_login | DateTime | 可為空 | | 上次登入時間 | |
| is\_superuser | BooleanField |  | | 是否為超級管理員 | |
| username | CharField | 150,唯一 | | 使用者名稱 | |
| first\_name | CharField | 150 | | 名字 | |
| last\_name | CharField | 150 | | 姓氏 | |
| email | EmailField | 唯一 | | 電子郵件 | |
| is\_staff | BooleanField |  | | 是否為後台管理員 | |
| is\_active | BooleanField |  | | 帳號是否啟用 | |
| date\_joined | DateTime |  | | 註冊時間 | |
| created\_at | DateTime |  | | 創建時間（你自訂的欄位） | |
| updated\_at | DateTime |  | | 更新時間（你自訂的欄位） | |
| groups | ManyToMany |  | | 所屬群組（權限用） | |
| user\_permissions | ManyToMany |  | | 個別權限 | |

1. metroline（捷運路線）

▼表 8-2‑3：metroline

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 捷運路線 | 資料表編號 | | 02 | |
| 英文名稱 | metroline | 主索引 | | id | |
| 資料檔陳述 | 儲存台北捷運的各路線 | | | | |
| 欄位名稱 | 型態 | | 長度/限制 | | 說明 |
| id | AutoField | |  | | 主鍵 |
| name | CharField | | 100,唯一 | | 捷運線名稱 |
| color | CharField | | 20 | | 捷運線顏色 |
| created\_at | DateTime | |  | | 創建時間 |
| updated\_at | DateTime | |  | | 更新時間 |

1. station（捷運站）

▼表 8-2‑4：station

|  |  |  |  |
| --- | --- | --- | --- |
| 中文名稱 | 捷運站 | 資料表編號 | 03 |
| 英文名稱 | station | 主索引 | id |
| 資料檔陳述 | 儲存台北捷運的各站站名 | | |
| 欄位名稱 | 型態 | 長度/限制 | 說明 |
| id | AutoField |  | 主鍵 |
| name | CharField | 100 | 站名 |
| metro\_line\_id | ForeignKey | MetroLine | 所屬捷運線 |
| latitude | DecimalField | 15,12 | 緯度 |
| longitude | DecimalField | 15,12 | 經度 |
| station\_code | CharField | 10 | 站點代碼 |
| created\_at | DateTime |  | 創建時間 |
| updated\_at | DateTime |  | 更新時間 |

1. restaurant（餐廳）

▼表 8-2‑5：restaurant

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 中文名稱 | 餐廳 | | 資料表編號 | | 04 |
| 英文名稱 | restaurant | | 主索引 | | id |
| 資料檔陳述 | 儲存餐廳名稱及詳細資料 | | | | |
| 欄位名稱 | 型態 | 長度/限制 | | 說明 | |
| id | AutoField |  | | 主鍵 | |
| name | CharField | 200 | | 店家名稱 | |
| address | CharField | 500 | | 地址 | |
| latitude | DecimalField | 15,12 | | 緯度 | |
| longitude | DecimalField | 15,12 | | 經度 | |
| phone | CharField | 20,可為空 | | 電話 | |
| google\_place\_id | CharField | 100,唯一 | | GooglePlacesID | |
| rating | DecimalField | 3,1,可為空 | | Google評分 | |
| price\_level | IntegerField | 可為空 | | 價格等級 | |
| website | URLField | 可為空 | | 網站 | |
| image | CharField | 500,可為空 | | 照片參考 | |
| created\_at | DateTime |  | | 創建時間 | |
| updated\_at | DateTime |  | | 更新時間 | |
| last\_review\_update | DateTime | 可為空 | | 最後評論更新時間 | |
| ... | ManyToMany | Station,FoodCategory | | 關聯捷運站/美食分類 | |

1. businesshours（餐廳營業時間）

▼表 8-2‑6：businesshours

|  |  |  |  |
| --- | --- | --- | --- |
| 中文名稱 | 餐廳營業時間 | 資料表編號 | 05 |
| 英文名稱 | businesshours | 主索引 | id |
| 資料檔陳述 | 儲存餐廳營業資訊 | | |
| 欄位名稱 | 型態 | 長度/限制 | 說明 |
| id | AutoField |  | 主鍵 |
| restaurant\_id | ForeignKey | Restaurant | 餐廳 |
| day\_of\_week | IntegerField |  | 星期幾 |
| open\_time | TimeField |  | 開店時間 |
| close\_time | TimeField |  | 關店時間 |
| is\_closed | BooleanField |  | 是否休息 |
| created\_at | DateTime |  | 創建時間 |
| updated\_at | DateTime |  | 更新時間 |

1. review（餐廳評論）

▼表 8-2‑7：review

|  |  |  |  |
| --- | --- | --- | --- |
| 中文名稱 | 餐廳評論 | 資料表編號 | 06 |
| 英文名稱 | review | 主索引 | id |
| 資料檔陳述 | 儲存Google的評論 | | |
| 欄位名稱 | 型態 | 長度/限制 | 說明 |
| id | AutoField |  | 主鍵 |
| restaurant\_id | ForeignKey | Restaurant | 餐廳 |
| user\_id | ForeignKey | User,可為空 | 使用者 |
| reviewer\_name | CharField | 100,可為空 | 評論者名稱 |
| rating | IntegerField |  | 評分 |
| comment | TextField | 可為空 | 評論內容 |
| created\_at | DateTime |  | 創建時間 |
| updated\_at | DateTime |  | 更新時間 |

1. checkinreview（打卡評論）

▼表 8-2‑8：checkinreview

|  |  |  |  |
| --- | --- | --- | --- |
| 中文名稱 | 打卡評論 | 資料表編號 | 07 |
| 英文名稱 | checkinreview | 主索引 | id |
| 資料檔陳述 | 儲存打卡任務的評論內容 | | |
| 欄位名稱 | 型態 | 長度/限制 | 說明 |
| id | AutoField |  | 主鍵 |
| restaurant\_name | CharField | 100 | 餐廳名稱 |
| metro\_line | CharField | 50 | 捷運線 |
| reviewer\_name | CharField | 50,預設匿名 | 評論者名稱 |
| rating | IntegerField |  | 評分 |
| comment | TextField | 可為空 | 評論內容 |
| created\_at | DateTime |  | 創建時間 |
| updated\_at | DateTime |  | 更新時間 |

1. traininfo（列車到站資訊）

▼表 8-2‑9：traininfo

|  |  |  |  |
| --- | --- | --- | --- |
| 中文名稱 | 列車到站資訊 | 資料表編號 | 08 |
| 英文名稱 | traininfo | 主索引 | id |
| 資料檔陳述 | 儲存捷運詳細到站資訊 | | |
| 欄位名稱 | 型態 | 長度/限制 | 說明 |
| id | AutoField |  | 主鍵 |
| train\_number | CharField | 50,可為空 | 列車號碼 |
| station\_name | CharField | 100 | 站名 |
| destination\_name | CharField | 100 | 目的地 |
| count\_down | CharField | 50 | 倒數時間 |
| now\_date\_time | DateTime |  | 當前時間 |
| created\_at | DateTime |  | 創建時間 |

1. stationid（車站ID對照表）

▼表 8-2‑10：stationid

|  |  |  |  |
| --- | --- | --- | --- |
| 中文名稱 | 車站ID對照表 | 資料表編號 | 09 |
| 英文名稱 | stationid | 主索引 | id |
| 資料檔陳述 | 儲存捷運路線名稱及站點資訊 | | |
| 欄位名稱 | 型態 | 長度/限制 | 說明 |
| id | AutoField |  | 主鍵 |
| station\_name | CharField | 100 | 車站名稱 |
| line\_name | CharField | 100 | 線路名稱 |
| station\_id | CharField | 20,唯一 | 車站ID |

# 第9章　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)－網頁介面撰寫

|  |  |  |
| --- | --- | --- |
| HTML－網頁介面撰寫 | | |
| 編號 | 檔案名稱 | 功能 |
| 1-1-1 | login.html | 註冊及登入 |
| 1-1-2 | homepage.html | 系統首頁 |
| 1-1-3 | night.html | 夜貓子專區 |
| 1-1-4 | drinks.html | 喝酒專區 |
| 1-1-5 | bars.html | 酒吧&夜店 |
| 1-1-6 | food\_checkin.html | 打卡任務 |
| 1-1-7 | food\_mission.html | 美食任務版 |
| 1-1-8 | store\_details.html | 美食任務版－店家詳細資訊 |
| 1-1-9 | food\_map.html | 美食地圖 |
| 1-1-10 | notify.html | 通知 |
| 1-1-11 | Member\_center.html | 會員中心 |
| 1-1-12 | edit\_profile.html | 修改會員資料 |
| 1-1-13 | favorites\_food.html | 收藏美食 |
| 1-1-14 | review.html | Google評論及地圖 |

▼表 9-1‑2：元件清單與規格描述表(前端)－網頁介面設計

|  |  |  |
| --- | --- | --- |
| CSS－網頁介面設計 | | |
| 編號 | 檔案名稱 | 功能 |
| 1-2-1 | login.css | 註冊及登入 |
| 1-2-2 | homepage.css | 系統首頁 |
| 1-2-3 | night.css | 夜貓子專區 |
| 1-2-4 | bars.css | 酒吧&夜店 |
| 1-2-5 | food\_checkin.css | 打卡任務 |
| 1-2-6 | food\_mission.css | 美食任務版 |
| 1-2-7 | store\_details.css | 美食任務版－店家詳細資訊 |
| 1-2-8 | food\_map.css | 美食地圖 |
| 1-2-9 | notify.css | 通知 |
| 1-2-10 | Member\_center.css | 會員中心 |
| 1-2-11 | edit\_profile.css | 修改會員資料 |
| 1-2-12 | favorites\_food.css | 收藏美食 |
| 1-2-13 | review.css | Google評論及地圖 |

▼表 9-1‑3：元件清單與規格描述表(前端)－網頁動畫設計

|  |  |  |
| --- | --- | --- |
| JavaScript－網頁動畫設計 | | |
| 編號 | 檔案名稱 | 功能 |
| 1-3-1 | food\_map.js | 依選捷運站載入餐廳，補Google照片，計算站→店距離與步行時間，支援收藏與通知，提供導航/評論連結 |
| 1-3-2 | station\_name.js | 動態建立「依路線分組且帶線色樣式」的捷運站下拉選單，可取得所選站的站碼與站名，供後續載入美食、算距離與導航使用 |
| 1-3-3 | checkin.js | 提供餐廳即時搜尋（Awesomplete+後端API）、可點選1～5星評分、依選取餐廳自動帶出捷運站名、以localStorage的username（無則顯示「訪客」）組成打卡資料並POST至/api/api/checkin-reviews/，含任務頁與首頁導向 |
| 1-3-4 | detail.js | 用Google Map Places定位並帶出評分、評論數、營業時間與照片；從後端載入打卡評論與關鍵字、備援地址；登入者的自家評論顯示會員名稱與頭像，並提供開啟Google評論、訂位、分享快捷 |
| 1-3-5 | mission.js | 載入熱門餐廳 → 用Google Places補圖資與營業資訊 → 計算距離最近捷運站與步行時間並提供導航 → 支援收藏與通知、顯示評分/評論與詳情連結 |
| 1-3-6 | stationdata.js | 站名自動完成＋交換、功能鍵導頁；Top10餐廳補照片與最近站距離/導航；兩欄彩色路線卡顯示查詢結果；首頁顯示台北車站即時列車 |
| 1-3-7 | drink.js | 動態產生捷運站下拉選單，點「查詢」即呼叫後端API取得該站末五班車時刻並渲染於頁面 |
| 1-3-8 | reviews.js | 從URL暫存取店家，用Google Places拉詳情與前5評論並標注地圖；若無id則Text Search以「相似度＋距離」挑最佳再呈現 |
| 1-3-9 | night.js | 整合夜貓美食＋末五班車：依站名篩餐廳、地圖定位＋照片快取、彈窗顯示評分/營業時間/距離並一鍵導航；同頁以站名查詢並呈現該站「末五班車」時刻 |

▼表 9-1‑4：元件清單與規格描述表(後端)－爬蟲

|  |  |  |
| --- | --- | --- |
| Python－爬蟲 | | |
| 編號 | 檔案名稱 | 功能 |
| 2-1-1 | import\_restaurants.py | 爬取餐廳資訊 |
| 2-1-2 | google\_maps\_reviews\_crawler.py | 爬取餐廳評論 |

▼表 9-1‑5：元件清單與規格描述表(後端)－讀取API

|  |  |  |
| --- | --- | --- |
| Python－讀取API | | |
| 編號 | 檔案名稱 | 功能 |
| 2-2-1 | urls.py | 定義捷運、餐廳、評論、酒吧、通知等API路由對應至views |
| 2-2-2 | views.py | 提供主要API，包含餐廳查詢、評論CRUD、打卡評論、捷運、路線、酒吧、通知等 |
| 2-2-3 | models.py | 定義資料模型，包含Restaurant、Review、CheckinReview、MetroFirstLastTrain等 |
| 2-2-4 | serializers.py | 序列化模型資料（部分API使用） |

▼表 9-1‑6：元件清單與規格描述表(後端)－Service

|  |  |  |
| --- | --- | --- |
| Python－Service | | |
| 編號 | 檔案名稱 | 功能 |
| 2-3-1 | review\_service.py | 建立餐廳、打卡評論、回算餐廳評分、同步Google Places評分與評論 |
| 2-3-2 | metro\_train\_service.py | 以TDX OAuth2取token，同步台北、新北首末班車資料並寫入資料庫 |

▼表 9-1‑7：元件清單與規格描述表(後端)－定時更新資訊

|  |  |  |
| --- | --- | --- |
| Python－定時更新資訊 | | |
| 編號 | 檔案名稱 | 功能 |
| 2-4-1 | update\_reviews.py | 批次更新Google Places餐廳評論與評分 |
| 2-4-2 | tasks.py | 背景任務、排程（若啟用APScheduler/Crontab） |

▼表 9-1‑8：元件清單與規格描述表(後端)－使用者資訊管理

|  |  |  |
| --- | --- | --- |
| Python－使用者資訊管理 | | |
| 編號 | 檔案名稱 | 功能 |
| 2-5-1 | urls.py | 帳號相關路由（登入、註冊、個資） |
| 2-5-2 | views.py | 帳號API，包含登入、註冊、取得使用者資訊 |
| 2-5-3 | models.py | 自訂使用者欄位（暱稱、生日、頭像等） |

▼表 9-1‑9：元件清單與規格描述表(後端)－串接相關網頁

|  |  |  |
| --- | --- | --- |
| Python－使用者資訊管理 | | |
| 編號 | 檔案名稱 | 功能 |
| 2-6-1 | urls.py | 專案全域路由入口，掛載各app的urls |
| 2-6-2 | settings.py | 專案設定：資料庫、APP、CORS、REST Framework、金鑰等 |

## 其他附屬之各種元件

▼表 9-2‑1：虛擬碼－homepage.html、homepage.css

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 1-1-2、1-2-2 | 程式名稱 | homepage.html、homepage.css |
| 目的 | 建立首頁頁面，提供捷運站搜尋、美食地圖導向、熱門店家顯示，以及RWD響應式版面設計。 | | |
| 虛擬碼 | | | |
| 1. 頁面基本結構設定  * 採用HTML 5標準結構，語系設定為zh-TW * head區： * 設定UTF-8編碼與viewport響應式設定 * 引入favicon（圖片/logo.png）、CSS樣式表static/css/homepage.css * 載入Awesomplete外部套件（自動完成搜尋）  1. 主要區塊說明  * header區塊 * 顯示網站Logo與標題「MEI食不打烊Taipei Metro」 * 使用.title-with-logo排版，支援窄螢幕自動換行 * 搜尋列 * 兩個輸入框（出發站、目的站）＋「交換」按鈕＋「查詢」按鈕 * 使用Awesomplete套件實現捷運站自動完成功能 * 採用.search-bar容器，支援RWD * 功能按鈕區 * 包含三個主要功能按鈕：夜貓子專區、打卡任務、捷運時刻表 * 使用.features容器，按鈕有icon與文字說明 * TOP 10店家區 * 使用.top-shops容器顯示當月熱門店家 * .shop-list為橫向可滑動區域 * .shop-item顯示單一店家資訊，包含圖片與文字 * 固定底部導覽列 * 使用<nav class="footer">固定在頁面底部 * 提供「首頁、美食地圖、通知、會員中心」4個導覽按鈕 * 自動根據網址高亮目前頁面  1. CSS設定重點  * RWD響應式版心 * 使用:root { --app-max: ... }控制版心寬度 * 根據螢幕寬度分別調整手機、平板、桌機版心最大寬度 * 全域樣式 * 使用box-sizing: border-box;統一計算方式 * body採用置中排列，背景為淺灰色 * 功能按鈕區 * display: flex; justify-content: space-around; * 小螢幕時按鈕允許換行 * TOP 10店家 * 使用橫向scroll-snap技術，使滑動更順暢 * 移除scrollbar以提升外觀 * Footer導覽列 * 使用position: fixed;固定在畫面底部 * 寬度與版心同步，支援iPhone safe area * 導覽按鈕在當前頁面時會加上.active樣式變白底  1. JavaScript功能  * 監聽DOMContentLoaded事件，自動為當前頁面對應的導覽按鈕加上active 類別 * 使用Awesomplete載入stationdata.js提供站名資料 * 載入Google Maps Places API（key已設定）  1. 頁面行為流程  * 使用者進入首頁 → 顯示標題、搜尋區、功能區 * 使用者輸入出發站、目的站 → Awesomplete自動完成捷運站名 * 點擊「查詢」按鈕 → 顯示查詢結果區域 * 下方固定導覽列可切換其他功能頁面（地圖、通知、會員中心） | | | |

▼表 9-2‑2：虛擬碼－Member\_center.html、Member\_center.css

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 1-1-11、1-2-10 | 程式名稱 | Member\_center.html、Member\_center.css |
| 目的 | 提供會員使用者查看與編輯個人資料、上傳頭像、快速進入收藏、美食地圖與捷運資訊，並支援RWD響應式版面。 | | |
| 虛擬碼 | | | |
| 1. 頁面基本結構設定  * 採用HTML 5標準結構，語系設定為zh-TW * <head>：設定favicon、引入CSS樣式、內嵌部分彈窗樣式 * <body>：以.container作為主版心，內容依序為： * 頁面標題 * 個人資料卡（含頭像） * 功能清單區 * 固定底部導覽列 * 編輯個人資料的彈窗（modal） * RWD設定 * 使用CSS變數--app-max控制不同裝置的版心寬度（手機480px、平板720px、桌機960px、超寬螢幕1200px） * 透過@media斷點調整間距與字體大小，確保小螢幕排版正常  1. 主要區塊說明  * 頁面標題區 * 顯示「會員中心」標題 * 使用.member-title加粗字體與固定左右padding，使標題與個資區對齊 * 個人資料卡 * 左側為頭像.avatar-wrap：採圓形容器，支援頭像圖片，右下角浮動相機按鈕（.avatar-edit），可觸發檔案選擇上傳頭像 * 右側為文字.profile-info：顯示會員名稱、Email、生日、電話、性別，「編輯資訊」按鈕會開啟彈窗 * 功能清單區 * 採垂直排列：「收藏美食」→ favorites\_food.html、「捷運時刻表」→ 外部連結、「捷運末5班」→ night.html、「登出」按鈕 * 使用.menu-item連結樣式，底部以分隔線區分項目 * 固定底部導覽列 * 使用position: fixed固定於畫面底部，寬度隨版心縮放 * 提供「首頁、美食地圖、通知、會員中心」4個導覽按鈕 * 自動根據網址高亮目前頁面 * 編輯個人資料彈窗 * 使用.modal-backdrop作為半透明背景 * .modal內包含Email、姓名、性別、生日、電話等輸入欄位 * 「取消」與「儲存」按鈕分別綁定關閉與送出更新API  1. CSS設定重點  * 採用多組:root變數管理尺寸（例如頭像大小、間距、按鈕位置） * .avatar-wrap使用overflow: visible讓相機按鈕能「浮在頭像外側」 * .footer透過transform: translateX(-50%)置中，並支援RWD * .menu-item移除超連結底線，並保持全區塊可點擊  1. JavaScript功能  * 登入檢查與提示 * checkLogin()：向後端/api/accounts/user/發送請求，確認是否登入 * 若未登入，顯示頂部提示條「尚未登入」，並附登入連結 * 載入個資 * loadProfile()：登入後自動從/api/accounts/profile/取回會員資料，填入資料卡與彈窗欄位 * 編輯與更新個資 * updateProfile()：將彈窗輸入的資料透過PUT請求送到/api/accounts/profile/ * 頭像上傳 * uploadAvatar(file)：將選擇的圖片以FormData送至/api/accounts/profile/avatar/，並即時更新頭像顯示 * Modal控制 * openModal() / closeModal()控制彈窗顯示與關閉 * 點擊背景可關閉彈窗。 * 登出功能 * 點擊「登出」按鈕，呼叫/api/accounts/logout/，並清除localStorage後跳轉login.html。  1. 頁面行為流程  * 使用者進入會員中心 → 頁面觸發loadProfile() → 檢查登入狀態 * 若已登入 → 載入並顯示個資（頭像、Email、電話等） * 使用者可： * 點擊相機圖示 → 選擇圖片 → 即時上傳更換頭像 * 點擊「編輯資訊」 → 開啟彈窗 → 修改資料 → 儲存更新 * 點擊底部導覽列按鈕可切換其他頁面 * 點擊「登出」後跳轉至登入頁  1. 特殊說明  * 後端API伺服器網址固定為http://140.131.115.112:8000 * 跨網域情況下，若後端開啟CSRF驗證，需設定CORS + CSRF * 所有資料請求皆使用credentials: 'include'以攜帶登入Cookie | | | |

▼表 9-2‑3：虛擬碼－night.html、night.css

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 1-1-3、1-2-3 | 程式名稱 | night.html、night.css |
| 目的 | 提供使用者在深夜時段，快速查找捷運站周邊的推薦美食，並查詢該站的末五班捷運車次資訊，同時串接Google Map顯示餐廳位置。 | | |
| 虛擬碼 | | | |
| 1. 頁面基本結構設定  * 採用HTML 5標準結構，語系設定為zh-TW * <head>： * 設定網頁標題與favicon * 引入外部CSS：夜貓/night.css * <body>：主要內容包含： * 返回首頁按鈕 * 「喝酒專區」膠囊按鈕 * 頁首（Logo、標題、副標題、搜尋列） * 推薦美食橫向滑動卡片區 * Google Map區塊 * 捷運末班車查詢區塊 * 尾端載入JS 與Google Maps API  1. 主要區塊說明  * 返回按鈕 & 喝酒專區 * 返回按鈕：固定在.container左上角，使用絕對定位，返回homepage.html * 喝酒專區按鈕：右上角膠囊風格連結（pill-link），導向drinks.html * 頁首區 * Logo圖示左上，使用絕對定位置於header區塊 * 主標題「MEI食不打烊」及副標題「夜貓子專區」置於右方，對齊右側 * 搜尋列：圓角白底，左側為搜尋圖示，右側為輸入欄，供使用者輸入關鍵字搜尋捷運站附近美食 * 推薦美食區 * 使用.recommend-container水平滑動容器，內含9 張.food-card 卡片 * 每張卡片包含：店家圖片、店名 + 商圈介紹、五星評價圖示、「MORE」按鈕 * 設計為手機橫向滑動瀏覽，桌機則可同時顯示多張卡片 * Google地圖區塊 * 預留.map容器，並使用Google Maps JavaScript API載入地圖 * 將來可搭配餐廳資料在地圖上顯示marker標記 * 捷運末班車查詢區 * 提供下拉選單選擇車站 * 「確定」按鈕觸發查詢 * 查詢結果（末五班車）會顯示於.late-info容器內  1. CSS設定重點  * 採用CSS變數--app-max控制版心最大寬度，並使用多段media query： * 手機：480px * 平板：720px * 桌機：960px * 大螢幕：1200px * .container使用position: relative以便內部按鈕與Logo絕對定位 * RWD響應式設計： * .recommend-container可橫向滑動以節省小螢幕空間 * .pill-link按鈕在小螢幕時縮小padding * .food-card採用scroll-snap技術讓滑動停留在卡片中央  1. JavaScript功能  * 登入檢查 * DOMContentLoaded時檢查localStorage中的isLoggedIn * 若未登入 → alert('請先登入！') → 導向login.html * Google Maps載入 * 使用Google Maps API Key與initMap() callback載入地圖 * 夜貓專區資料 * 透過night.js可以補上餐廳資料抓取、末班車查詢API串接與地圖標記  1. 頁面行為流程  * 使用者進入夜貓子專區 → 檢查登入 → 若未登入跳轉至登入頁 * 若登入： * 顯示推薦美食清單 * 載入Google地圖 * 使用者可： * 在搜尋欄輸入站名或關鍵字查詢 * 水平滑動瀏覽推薦美食 * 點擊「MORE」開啟餐廳詳細資料 * 選擇車站並查詢末班車 → 結果顯示在.late-info * 可透過右上膠囊按鈕進入喝酒專區，或點擊返回按鈕回首頁 | | | |

▼表 9-2‑4：虛擬碼－missiom.js

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 1-3-5 | 程式名稱 | missiom.js |
| 目的 | 在任務頁載入「熱門打卡餐廳」，計算距離最近捷運站並生成卡片（評分/營業/地址），提供導航與收藏（含通知），並用Google Places自動補照片與細節。 | | |
| 虛擬碼 | | | |
| 1. 基本設定  * 通知模組：notifications\_v1以localStorage保存收藏/取消收藏訊息（id、時間、內容、目標店家） * 收藏模組：fav\_restaurants\_v1儲存餐廳（以name@address鍵去重），toggleFav()切換並更新按鈕樣式 * 站點快取/距離：啟動時抓取所有路線與站點座標為MRT\_STATIONS；Haversine算直線距離，WALK\_M\_PER\_MIN＝80換算步行分鐘 * 均消換算：Google price\_level → 顯示區間（$100–200…$600–1000＋）  1. 資料取得與清洗  * 呼叫/api/api/top-checkin-restaurants/取熱門餐廳 * 過濾測試資料，normalize()统一欄位：name/rating/review\_count/address/website/phone/opening\_text/price\_text/lat/lng  1. 卡片渲染（Cards）  * 每筆輸出：圖片（預設占位）、店名、評分＋評論數、營業時間、地址、距離列、導航與收藏按鈕、標籤（詳情/均消/官網） * 距離列文案：🚶 距離{最近捷運站}站約X公尺（步行約Y分鐘） * 導航：有座標 → 以lat,lng開Google Maps；無則以地址查詢 * 收藏：即時切換文字「收藏/已收藏」，並寫入通知列  1. Google Places增強（漸進補強）  * textSearch → getDetails：取得照片、完整地址、今日營業字串、price\_level、更精準座標。 * 只更新需要的欄位： * 圖片：替換卡片背景圖 * 地址/營業時間/均消：覆寫空白或占位值 * 座標：更新後重新計算距離 * 收藏狀態依最新payload重新判定  1. 例外處理與回退  * 後端或Places失敗：保持占位圖與「—」文案，距離列隱藏但卡片與導航仍可用 * 空清單與錯誤都有明確提示訊息  1. 擴充、介面連結  * 「查看詳情」連至store\_details.html?name=...可銜接評論或地圖詳頁 * 保留FAV\_PAGE常數，之後可導向「我的收藏」 * 站點資料若於前端補上lat/lng，可跳過首次API並提升距離精度 | | | |

▼表 9-2‑5：虛擬碼－stationdata.js

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 1-3-6 | 程式名稱 | stationdata.js |
| 目的 | 在首頁整合「捷運站搜尋與路線查詢、Top10美食卡片（含距離/導航）、即時列車資訊」，提供快速找店與規劃動線的流暢體驗。 | | |
| 虛擬碼 | | | |
| 1. 基本設定  * 站點資料：內建stationData（BR/R/G/O/BL五線站碼與站名），供自動完成與路線著色使用 * 快取：stationCoordCache（站名→座標）與內部MRT\_STATIONS（所有站座標）避免重複查詢 * 常數：步行換算WALK\_M\_PER\_MIN=80；Haversine距離函式；Google Maps導航URL產生器  1. 版位與互動  * 底部導覽高亮：依目前頁檔名自動替對應.nav-btn加上active * 功能按鈕（.feature-btn） * night.html夜貓美食 * food\_checkin.html打卡 * 外連「到站時間」亦支援data-link（站內）與data-external（外連）屬性覆寫 * TOP 10區 * 首先載入全線站座標（/api/lines/\*/stations/），建立MRT\_STATIONS * 取/api/api/restaurants/top10/ → 依評分排序取前10 * 卡片渲染：先畫出店名/評分/最近捷運站距離＋「導航」按鈕，再以Google Places只補圖（避免覆蓋既有距離/按鈕） * 距離文案：🚶 距離{站名}站約X公尺（步行約Y分） * 自動插入卡片CSS（行高、圖片比例、導航鈕樣式等） * 即時列車（首頁熱門站） * 預設查「台北車站」：呼叫/api/track-info/，彙整各線包含該站的列車，顯示出發站/目的地/方向/倒數與時間  1. 站名搜尋與輸入體驗  * Awesomplete自動完成：由stationData產生「站名(站碼)」清單；支援名稱與站碼同時比對 * 輸入上色：依站碼前綴（BR/R/G/O/BL）即時變更輸入框背景色（line color） * 交換起訖：#swap-btn互換兩欄內容；防呆避免起訖相同 * 解析器：parseInput / findBestStation支援「站名」或「站名(站碼)」與去尾「站」  1. 路線查詢（雙欄卡片）  * API：POST /api/api/metro-route/，body：{ start, end }（已清理成標準站名） * 方向推斷：依終點關鍵字粗估「東/西/南/北向」 * 渲染：beautifulRouteCard(result,{startName,endName,dir}) * 以result.Path拆站序，兩欄排列；上方顯示起/迄站與站碼、方向與預估時間 * 依相鄰站是否同線，用線色著色節點/線段；result.TransferStations以綠色環標「轉乘」 * RWD：<420px自動改為單欄 * 額外插入樣式使站名右移，避免貼線  1. Google Maps/Places使用  * Top10照片：textSearch取首筆照片URL，只更新<img>；未載入Places時用預設圖 * 導航：若有經緯度 → 一鍵開啟Google Maps步行導航；無座標改以地址查詢  1. 錯誤與保護  * Places未載入時僅顯示預設圖並記錄警告 * API/網路失敗時提供替代文案（Top10、即時列車、路線查詢） * 站名輸入不可相同；空值提示  1. 可擴充建議  * stationData補上lat/lng可免初次API取座標、提升距離精度 * Top10卡片加入「收藏」鈕（沿用既有收藏模組） * 即時列車加入站名切換，與首頁搜尋共用自動完成功能 | | | |

▼表 9-2‑6：虛擬碼－metro\_train\_service.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-3-2 | 程式名稱 | metro\_train\_service.py |
| 目的 | 整合台北捷運官方API，提供列車時刻表服務和資料同步功能。 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫： * requests：HTTP請求模組，用於發送API請求 * datetime：日期時間處理模組，用於時間格式轉換 * django.conf.settings：Django設定模組，讀取專案設定 * django.utils.timezone：時間處理模組，處理時區相關功能 * logging：日誌記錄模組，記錄操作日誌和錯誤訊息 * ..models：導入自定義模型，用於資料庫操作  1. API認證功能  * get\_access\_token： * 設定token\_url為TDX API認證端點 * 設定headers：Content-Type為application/x-www-form-urlencoded * 設定data：grant\_type為client\_credentials * 從settings 讀取TDX\_CLIENT\_ID和TDX\_CLIENT\_SECRET * 發送POST請求到認證端點 * 檢查回應狀態碼是否為200 * 解析JSON回應，提取access\_token * 處理認證失敗情況，記錄錯誤日誌 * 回傳有效的access token或None  1. 捷運時刻表獲取功能  * fetch\_metro\_first\_last\_trains： * 調用get\_access\_token獲取access token * 驗證access token是否有效 * 設定台北捷運API URL：TRTC端點 * 設定新北捷運API URL：NTMC端點 * 設定headers：Authorization為Bearer token * 發送GET請求到台北捷運API * 發送GET請求到新北捷運API * 解析兩個API的JSON回應 * 合併台北和新北的資料 * 初始化更新計數器update\_count = 0 * 遍歷所有資料項目進行處理  1. 資料處理和轉換功能  * 時間格式轉換： * 使用datetime.strptime解析FirstTrainTime字串 * 使用datetime.strptime解析LastTrainTime字串 * 轉換為Django TimeField格式 * 處理時區轉換：使用fromisoformat和replace('Z', '+00:00') * 資料驗證和清理： * 檢查必要欄位是否存在 * 驗證資料格式是否正確 * 處理缺失或無效資料 * 記錄處理錯誤的日誌  1. 資料庫更新功能  * update\_or\_create操作： * 使用MetroFirstLastTrain.objects.update\_or\_create() * 設定唯一識別欄位：line\_no、line\_id、station\_id、destination\_station\_id、train\_type * 設定defaults參數包含所有更新欄位 * 處理創建和更新兩種情況 * 增加更新計數器 * 處理資料庫操作錯誤 * 錯誤處理機制： * 使用try-except包裝每個資料處理操作 * 記錄詳細的錯誤日誌 * 繼續處理下一個資料項目 * 不因單一錯誤中斷整個流程  1. 站點查詢功能  * get\_station\_first\_last\_trains： * 接收station\_id參數 * 使用MetroFirstLastTrain.objects.filter(station\_id=station\_id)查詢 * 回傳查詢結果 * 處理查詢錯誤情況 * 記錄查詢操作日誌 * 回傳None如果查詢失敗  1. 日誌記錄功能  * 成功日誌： * 記錄成功更新的資料筆數 * 記錄API請求成功訊息 * 記錄資料庫操作成功訊息 * 錯誤日誌： * 記錄API認證失敗 * 記錄API請求失敗 * 記錄資料處理錯誤 * 記錄資料庫操作錯誤 * 包含詳細的錯誤訊息和堆疊追蹤 | | | |

▼表 9-2‑7：虛擬碼－models.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-2-3 | 程式名稱 | models.py |
| 目的 | 定義系統中所有資料庫模型和資料結構，建立完整的資料庫架構。 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫： * django.db.models：Django ORM核心模組，提供資料庫操作功能 * django.contrib.auth.models：Django用戶認證模組，提供用戶管理功能 * django.conf.settings：Django設定模組，讀取專案設定 * django.utils.timezone：時間處理模組，處理時區相關功能 * django.core.validators：資料驗證模組，提供欄位驗證功能  1. 用戶模型定義（User）  * 繼承AbstractUser： * 擴展Django內建用戶功能，保留原有認證機制 * 新增個人化欄位：display\_name、gender、birthday、phone\_number * 新增頭像上傳功能：avatar欄位，支援圖片上傳 * 設定email為唯一欄位，確保用戶唯一性 * 新增時間戳記：created\_at、updated\_at自動記錄時間  1. 捷運系統模型定義  * MetroLine模型： * 管理捷運線路基本資訊：name（線路名稱）、color（線路顏色） * 設定name為唯一欄位，避免重複線路 * 新增時間戳記：created\_at、updated\_at * 定義\_\_str\_\_方法，回傳線路名稱 * Station模型： * 管理捷運站點詳細資訊：name、latitude、longitude、station\_code * 與MetroLine建立外鍵關聯：metro\_line欄位 * 使用DecimalField精確儲存座標資訊 * 設定station\_code為唯一識別碼 * 新增時間戳記：created\_at、updated\_at  1. 餐廳系統模型定義  * Restaurant模型： * 管理餐廳完整資訊：name、address、phone、rating、price\_level * 整合Google Places API：google\_place\_id欄位 * 與Station建立多對多關聯：nearby\_stations欄位 * 與FoodCategory建立多對多關聯：categories欄位 * 新增評分相關欄位：rating、price\_level * 新增時間戳記：created\_at、updated\_at、last\_review\_update * FoodCategory模型： * 管理美食分類：name、description * 設定name為唯一欄位 * 新增時間戳記：created\_at、updated\_at * 定義Meta類別：verbose\_name、ordering * BusinessHours模型： * 管理餐廳營業時間：day\_of\_week、open\_time、close\_time、is\_closed * 與Restaurant建立外鍵關聯 * 定義DAY\_CHOICES選擇項目 * 設定unique\_together約束：restaurant + day\_of\_week  1. 評論系統模型定義  * Review模型： * 管理餐廳評論：reviewer\_name、rating、comment * 與Restaurant建立外鍵關聯：restaurant欄位 * 與User建立外鍵關聯：user欄位（可為空） * 設定評分範圍：1～5星評分 * 新增時間戳記：created\_at、updated\_at * 定義Meta類別：ordering按時間倒序 * CheckinReview模型： * 管理打卡評論：restaurant\_name、metro\_line、reviewer\_name、rating、comment * 支援匿名評論功能 * 新增時間戳記：created\_at、updated\_at * 設定db\_table自定義表格名稱  1. 互動功能模型定義  * CheckinReviewLike模型： * 管理評論按讚功能：review、user、created\_at * 與CheckinReview建立外鍵關聯 * 與User建立外鍵關聯 * 設定unique\_together約束：review + user * CheckinReviewFavorite模型： * 管理評論收藏功能：review、user、created\_at * 與CheckinReview建立外鍵關聯 * 與User建立外鍵關聯 * 設定unique\_together約束：review + user * Notification模型： * 管理通知系統：recipient、sender、notification\_type、title、content * 定義NOTIFICATION\_TYPES選擇項目：like、favorite、comment * 新增is\_read布林欄位 * 與CheckinReview建立外鍵關聯（可為空） * 新增時間戳記：created\_at、updated\_at | | | |

▼表 9-2‑8：虛擬碼－views.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-2-2 | 程式名稱 | views.py |
| 目的 | 提供完整的RESTful API端點，處理前端請求並回傳格式化的資料。 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫： * rest\_framework：Django REST Framework核心模組 * rest\_framework.viewsets：提供ViewSet類別 * rest\_framework.decorators：提供API裝飾器 * rest\_framework.response：提供Response類別 * rest\_framework.permissions：提供權限控制 * django.shortcuts：Django快捷函數 * django.core.cache：Django快取系統 * requests：HTTP請求模組 * json：JSON處理模組用戶模型定義（User）  1. ViewSet類別定義  * MetroLineViewSet： * 繼承ModelViewSet，提供完整CRUD操作 * 設定queryset = MetroLine.objects.all() * 設定serializer\_class = MetroLineSerializer * 設定permission\_classes = [AllowAny] * 支援GET/api/metro-lines/端點 * StationViewSet： * 繼承ModelViewSet，提供完整CRUD操作 * 設定queryset = Station.objects.all() * 設定serializer\_class = StationSerializer * 設定permission\_classes = [AllowAny] * 支援GET/api/stations/端點 * RestaurantViewSet： * 繼承ModelViewSet，提供完整CRUD操作 * 設定queryset = Restaurant.objects.all() * 設定serializer\_class = RestaurantSerializer * 設定permission\_classes = [AllowAny] * 支援GET/api/restaurants/端點  1. 餐廳查詢API函數  * get\_restaurants\_by\_station： * 使用@api\_view(['GET'])裝飾器 * 設定@permission\_classes([AllowAny]) * 從request.GET.get('station\_id')獲取站點ID * 驗證站點ID是否存在 * 使用Station.objects.get(id=station\_id)查找站點 * 使用Restaurant.objects.filter(nearby\_stations=station)查詢附近餐廳 * 使用RestaurantSerializer序列化資料 * 回傳Response(serializer.data) * 處理Station.DoesNotExist異常 * get\_restaurants\_by\_line： * 使用@api\_view(['GET'])裝飾器 * 從request.GET.get('line\_id')獲取線路ID * 使用MetroLine.objects.get(id=line\_id)查找線路 * 使用Station.objects.filter(metro\_line=line)查找線路所有站點 * 使用Restaurant.objects.filter(nearby\_stations\_\_in=stations).distinct()查詢沿線餐廳 * 使用RestaurantSerializer序列化資料 * 回傳Response(serializer.data) * search\_restaurants： * 使用@api\_view(['GET'])裝飾器 * 從request.GET.get('name')獲取搜尋關鍵字 * 驗證關鍵字是否存在 * 使用Restaurant.objects.filter(name\_\_icontains=name)進行模糊搜尋 * 準備回應資料，包含餐廳基本資訊和捷運站資訊 * 回傳格式化的搜尋結果  1. 評論管理API函數  * create\_restaurant\_review： * 使用@api\_view(['POST'])裝飾器 * 驗證必要欄位：rating * 檢查評分範圍：1～5 * 調用create\_review\_from\_frontend函數 * 處理創建失敗情況 * 回傳成功訊息和評論ID * create\_checkin\_review\_api： * 使用@api\_view(['POST'])裝飾器 * 驗證必要欄位：restaurant\_name、metro\_line、rating * 使用CheckinReview.objects.create()創建打卡評論 * 設定預設reviewer\_name為'匿名' * 回傳成功訊息和評論ID * get\_restaurant\_reviews： * 使用@api\_view(['GET'])裝飾器 * 從request.query\_params.get('restaurant\_name')獲取餐廳名稱 * 從request.query\_params.get('limit')獲取數量限制 * 使用CheckinReview.objects.filter()查詢評論 * 使用order\_by('-created\_at')按時間倒序排列 * 格式化回傳資料 * 回傳評論列表  1. 捷運資訊API函數  * get\_track\_info： * 使用@api\_view(['GET'])裝飾器 * 設定SOAP請求標頭：Content-Type、SOAPAction、Host * 從環境變數獲取API帳號密碼：METRO\_API\_USERNAME、METRO\_API\_PASSWORD * 驗證帳號密碼是否存在 * 建立SOAP XML請求格式 * 發送POST請求到台北捷運API * 檢查回應狀態碼 * 解析JSON回應資料 * 按照路線分組資料 * 回傳格式化的列車資訊 * get\_station\_last\_five\_trains： * 使用@api\_view(['GET'])裝飾器 * 從request.GET.get('station\_name')獲取站點名稱 * 驗證站點名稱是否存在 * 使用MetroLastFiveTrains.objects.filter()查詢末五班車 * 使用order\_by()按路線和方向排序 * 整理資料格式 * 回傳末五班車資訊  1. 互動功能API函數  * toggle\_checkin\_review\_like： * 使用@api\_view(['POST'])裝飾器 * 檢查用戶是否已登入 * 使用get\_object\_or\_404獲取評論 * 使用CheckinReviewLike.objects.get\_or\_create()處理按讚 * 如果已按讚則刪除，未按讚則創建 * 創建通知給評論作者 * 計算總讚數 * 回傳按讚狀態和數量 * toggle\_checkin\_review\_favorite： * 使用@api\_view(['POST'])裝飾器 * 檢查用戶是否已登入 * 使用get\_object\_or\_404獲取評論 * 使用CheckinReviewFavorite.objects.get\_or\_create()處理收藏 * 如果已收藏則刪除，未收藏則創建 * 創建通知給評論作者 * 計算總收藏數 * 回傳收藏狀態和數量 * get\_user\_notifications： * 使用@api\_view(['GET'])裝飾器 * 檢查用戶是否已登入 * 查詢未讀通知：Notification.objects.filter(recipient=user, is\_read=False) * 查詢所有通知：Notification.objects.filter(recipient=user) * 按時間倒序排列 * 格式化通知資料 * 回傳通知列表和未讀數量 | | | |

▼表 9-2‑9：虛擬碼－serializers.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-2-4 | 程式名稱 | serializers.py |
| 目的 | 將Django模型序列化為JSON格式，提供API資料轉換功能。 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫： * rest\_framework.serializers：Django REST Framework序列化器 * rest\_framework.fields：提供各種欄位類型 * 自定義模型：從models.py導入所有模型類別 * django.db.models：Django ORM模組  1. 基礎序列化器定義  * MetroLineSerializer： * 繼承ModelSerializer * 設定model = MetroLine * 定義fields = ['id', 'name', 'color', 'created\_at', 'updated\_at'] * 自動處理模型欄位序列化 * 支援建立和更新操作 * StationSerializer： * 繼承 ModelSerializer * 設定 model = Station * 定義 fields = ['id', 'name', 'latitude', 'longitude', 'station\_code', 'metro\_line', 'created\_at', 'updated\_at'] * 使用depth = 1自動序列化外鍵關聯 * 包含捷運線路資訊  1. 餐廳序列化器定義  * RestaurantSerializer： * 繼承ModelSerializer * 設定model = Restaurant * 定義fields = ['id', 'name', 'address', 'latitude', 'longitude', 'phone', 'rating', 'price\_level', 'website', 'image', 'nearby\_stations', 'categories', 'created\_at', 'updated\_at'] * 使用StringRelatedField序列化多對多關聯 * 新增計算欄位：average\_rating、review\_count * 使用SerializerMethodField計算平均評分 * 使用SerializerMethodField計算評論數量 * FoodCategorySerializer： * 繼承ModelSerializer * 設定model = FoodCategory * 定義fields = ['id', 'name', 'description', 'created\_at', 'updated\_at'] * 支援建立和更新操作  1. 評論序列化器定義  * ReviewSerializer： * 繼承ModelSerializer * 設定model = Review * 定義fields = ['id', 'restaurant', 'reviewer\_name', 'rating', 'comment', 'created\_at', 'updated\_at'] * 使用StringRelatedField序列化餐廳資訊 * 新增驗證：rating範圍1～5 * 支援建立和更新操作 * CheckinReviewSerializer： * 繼承ModelSerializer * 設定model = CheckinReview * 定義fields = ['id', 'restaurant\_name', 'metro\_line', 'reviewer\_name', 'rating', 'comment', 'created\_at', 'updated\_at'] * 新增驗證：rating範圍1～5 * 支援建立和更新操作  1. 互動功能序列化器定義  * CheckinReviewLikeSerializer： * 繼承ModelSerializer * 設定model = CheckinReviewLike * 定義fields = ['id', 'review', 'user', 'created\_at'] * 使用StringRelatedField序列化關聯物件 * 支援建立和刪除操作 * CheckinReviewFavoriteSerializer： * 繼承ModelSerializer * 設定model = CheckinReviewFavorite * 定義fields = ['id', 'review', 'user', 'created\_at'] * 使用StringRelatedField序列化關聯物件 * 支援建立和刪除操作 * NotificationSerializer： * 繼承ModelSerializer * 設定model = Notification * 定義fields = ['id', 'recipient', 'sender', 'notification\_type', 'title', 'content', 'is\_read', 'created\_at', 'updated\_at'] * 使用StringRelatedField序列化用戶資訊 * 支援更新已讀狀態  1. 捷運資訊序列化器定義  * MetroFirstLastTrainSerializer： * 繼承ModelSerializer * 設定model = MetroFirstLastTrain * 定義fields = ['id', 'line\_no', 'line\_id', 'station\_id', 'station\_name', 'station\_name\_en', 'trip\_head\_sign', 'destination\_station\_id', 'destination\_station\_name', 'destination\_station\_name\_en', 'train\_type', 'first\_train\_time', 'last\_train\_time', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'national\_holidays', 'src\_update\_time', 'update\_time', 'version\_id'] * 支援查詢和篩選操作 * MetroLastFiveTrainsSerializer： * 繼承ModelSerializer * 設定model = MetroLastFiveTrains * 定義fields = ['id', 'line\_no', 'line\_id', 'station\_id', 'station\_name', 'station\_name\_en', 'trip\_head\_sign', 'destination\_station\_id', 'destination\_station\_name', 'destination\_station\_name\_en', 'train\_type', 'train\_time', 'train\_sequence', 'monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'national\_holidays', 'src\_update\_time', 'update\_time', 'version\_id'] * 支援查詢和篩選操作  1. 自定義序列化器方法  * get\_average\_rating： * 計算餐廳平均評分 * 使用aggregate方法計算平均值 * 處理無評分情況 * 回傳格式化評分 * get\_review\_count： * 計算餐廳評論數量 * 使用count方法統計評論 * 回傳評論總數 * validate\_rating： * 驗證評分範圍 * 檢查評分是否在1～5範圍內 * 回傳驗證錯誤或通過 | | | |

▼表 9-2‑10：虛擬碼－review\_service.py

|  |  |  |  |
| --- | --- | --- | --- |
| 編號 | 2-3-1 | 程式名稱 | review\_service.py |
| 目的 | 處理評論相關業務邏輯，整合Google Places API提供評論同步功能。 | | |
| 虛擬碼 | | | |
| 1. 初始化與設置  * 載入所需的模組和庫： * requests：HTTP請求模組，用於發送API請求 * datetime：日期時間處理模組，用於時間格式轉換 * timedelta：時間差計算模組，用於時間比較 * django.conf.settings：Django設定模組，讀取專案設定 * django.utils.timezone：時間處理模組，處理時區相關功能 * django.db.models：Django ORM模組，用於資料庫操作 * django.db.models.Avg：聚合函數，用於計算平均值 * logging：日誌記錄模組，記錄操作日誌和錯誤訊息 * ..models：導入自定義模型，用於資料庫操作  1. 評論創建功能  * create\_review\_from\_frontend： * 接收restaurant\_id和review\_data參數 * 驗證評分範圍：檢查rating是否為1～5的整數 * 記錄無效評分的錯誤日誌 * 使用Restaurant.objects.get(id=restaurant\_id)查找餐廳 * 處理Restaurant.DoesNotExist異常 * 使用Review.objects.create()創建評論記錄 * 設定reviewer\_name預設值為 '匿名' * 設定comment預設值為空字串 * 設定created\_at為當前時間  1. 餐廳評分更新功能  * 更新餐廳統計資料： * 使用Review.objects.filter(restaurant=restaurant)查詢所有評論 * 使用reviews.count()計算評論總數 * 使用reviews.aggregate(avg\_rating=models.Avg('rating'))計算平均評分 * 更新restaurant.total\_ratings欄位 * 更新restaurant.rating欄位 * 使用restaurant.save()儲存變更 * 記錄成功創建評論的日誌  1. 打卡評論功能  * create\_checkin\_review： * 接收restaurant\_id和review\_data參數 * 驗證評分範圍：檢查rating是否為1～5的整數 * 記錄無效評分的錯誤日誌 * 使用Restaurant.objects.get(id=restaurant\_id)查找餐廳 * 處理Restaurant.DoesNotExist異常 * 使用CheckinReview.objects.create()創建打卡評論 * 設定reviewer\_name預設值為'匿名' * 設定comment預設值為空字串 * 設定created\_at為當前時間 * 記錄成功創建打卡評論的日誌  1. 評論查詢功能  * get\_checkin\_reviews： * 接收restaurant\_id和limit參數 * 使用CheckinReview.objects.filter(restaurant\_id=restaurant\_id)查詢評論 * 使用order\_by('-created\_at')按時間倒序排列 * 使用[:limit]限制回傳數量 * 格式化回傳資料為字典列表 * 包含id、reviewer\_name、rating、comment、created\_at欄位 * 使用strftime格式化時間字串 * 處理查詢錯誤情況 * 回傳空列表如果查詢失敗  1. Google Places API整合功能  * update\_restaurant\_reviews： * 查詢需要更新的餐廳：last\_review\_update為空或超過6小時 * 檢查是否有餐廳需要更新 * 記錄需要更新的餐廳數量 * 遍歷每個需要更新的餐廳 * 構建Google Places API請求URL * 設定請求參數：place\_id、fields、key、language * 發送GET請求到Google Places API * 檢查API回應狀態  1. Google Places API資料處理  * 解析API回應： * 檢查data['status']是否為'OK' * 記錄API請求失敗的警告日誌 * 更新餐廳評分：data['result']['rating'] * 更新評論總數：len(data['result'].get('reviews', [])) * 儲存餐廳變更 * 處理評論資料： * 遍歷data['result']['reviews']中的每個評論 * 檢查評論是否已存在：使用reviewer\_name和created\_at查詢 * 如果評論不存在則創建新記錄 * 使用datetime.fromtimestamp轉換時間戳記 * 創建Review記錄：restaurant、reviewer\_name、rating、comment、created\_at * 更新最後更新時間： * 設定restaurant.last\_review\_update為當前時間 * 儲存餐廳變更 * 記錄成功更新評論的日誌  1. 錯誤處理和日誌記錄  * 異常處理： * 使用try-except包裝每個操作 * 處理Restaurant.DoesNotExist異常 * 處理API請求異常 * 處理資料庫操作異常 * 記錄詳細的錯誤日誌 * 日誌記錄： * 記錄成功操作的訊息 * 記錄錯誤操作的詳細資訊 * 包含相關的參數和錯誤訊息 * 使用適當的日誌等級（info、warning、error） | | | |

# 第10章　測試模型

## 測試計畫

本系統所需的硬體為瀏覽器iOS全系列及Chrome 140.0.7339以上的版本。在系統程式開發設計完成後，我們將程式存在虛擬機後端後於手機上網頁進行測試，確認每項功能是否能夠執行成功。

* 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. 更改會員資料：修改使用者儲存的最新個人資料。
  4. 站點查詢：查詢站點之間的路線資訊。
  5. 當月TOP 10店家：推薦當月評論星數前10的店家資訊。
  6. 夜貓子專區：夜間使用者以捷運站名尋找美食及捷運末班車資訊。
  7. 喝酒專區：酒吧和酒店營業詳細資訊及代駕服務。
  8. 打卡任務評論：透過打卡任務來蒐集使用者的評論內容。
  9. 美食任務版：查詢所有店家的評論及營業詳細資訊。
  10. 捷運時刻表：提供北捷各站點詳細到站時間。
  11. 美食地圖：以捷運站名查詢各站周邊美食資訊。
  12. 收藏及通知：收藏美食後會通知收藏的相關訊息。

## 測試個案與測試結果

▼表 10-2‑1：測試結果－會員註冊

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (1) | 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會員資料 3. 若無重複的會員資料則直接註冊該筆會員資料 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑2：測試結果－會員登入

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (2) | 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. 驗證資料庫是否有重複的會員資料 3. 若查無符合的會員資料無法登入，需先註冊帳號 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑3：測試結果－更改會員資料

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (3) | 功能名稱 | 更改會員資料 |
| 測試目標 | 修改使用者儲存的最新個人資料 | | |
| 測試作業 | 1. 點選【會員中心】頁面中的編輯資訊 2. 是否能夠更改會員資料並儲存 3. 再次查看會員資料是否為更改過的資料 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑4：測試結果－站點查詢

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (4) | 功能名稱 | 站點查詢 |
| 測試目標 | 查詢站點之間的路線資訊 | | |
| 測試作業 | 1. 點選【首頁】頁面 2. 站點選單點選後按下查詢 3. 查看站點與路線的結果 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑5：測試結果－當月TOP 10店家

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (5) | 功能名稱 | 當月TOP 10店家 |
| 測試目標 | 推薦當月評論星數前10的店家資訊 | | |
| 測試作業 | 1. 點選【首頁】頁面 2. 滑至最下方 3. 查看店家詳細資訊 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑6：測試結果－夜貓子專區

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (6) | 功能名稱 | 夜貓子專區 |
| 測試目標 | 夜間使用者以捷運站名尋找美食及捷運末班車資訊 | | |
| 測試作業 | 1. 點選【夜貓子專區】頁面 2. 搜尋欄點選欲查詢站名 3. 查看站點周邊夜間美食詳細資訊 4. 選擇欲查詢的站名末五班車 5. 查看末五班車資訊 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑7：測試結果－喝酒專區

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (7) | 功能名稱 | 喝酒專區 |
| 測試目標 | 酒吧和酒店營業詳細資訊及代駕服務 | | |
| 測試作業 | 1. 點選【夜貓子專區】頁面 2. 點選右上角【喝酒專區】按鈕 3. 點選【】進入酒吧&夜店查詢頁面 4. 選擇欲查看站點周邊酒吧&夜店營業詳細資訊 5. 點選【】進入代駕頁面即跳轉至Uber 6. 輸入上車地及下車地 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑8：測試結果－打卡任務評論

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (8) | 功能名稱 | 打卡任務評論 |
| 測試目標 | 透過打卡任務來蒐集使用者的評論內容 | | |
| 測試作業 | 1. 點選【首頁】頁面 2. 點選【打卡任務】按鈕 3. 依序輸入相對應的資料後按下送即可完成 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑9：測試結果－美食任務版

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (9) | 功能名稱 | 美食任務版 |
| 測試目標 | 查詢所有店家的評論及營業詳細資訊 | | |
| 測試作業 | 1. 點選【打卡任務】按鈕 2. 點選中間【美食任務版】按鈕 3. 即可查看店家的相關資訊，也可看到其他用戶的評論 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑10：測試結果－捷運時刻表

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (10) | 功能名稱 | 捷運時刻表 |
| 測試目標 | 提供北捷各站點詳細到站時間 | | |
| 測試作業 | 1. 點選【捷運時刻表】按鈕 2. 跳轉至北捷各路線詳細站點到站時間 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑11：測試結果－美食地圖

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (11) | 功能名稱 | 美食地圖 |
| 測試目標 | 以捷運站名查詢各站周邊美食資訊 | | |
| 測試作業 | 1. 點選【美食地圖】頁面 2. 選擇欲查詢站點名稱，即可出現該站TOP 3美食資訊 3. 如要查看更多該站美食，可下滑點選【查看更多美食】 | | |
| 測試結果 | 成功 | | |

▼表 10-2‑12：測試結果－收藏及通知

|  |  |  |  |
| --- | --- | --- | --- |
| 功能編號 | (12) | 功能名稱 | 收藏及通知 |
| 測試目標 | 收藏美食後會通知收藏的相關訊息 | | |
| 測試作業 | 1. 在【美食任務版】、【美食地圖】點選收藏按鈕後會跳通知在【通知】頁面 2. 點選【會員中心】頁面選擇【收藏美食】按鈕 3. 該頁面會出現先前收藏的店家資訊 | | |
| 測試結果 | 成功 | | |

# 第11章　操作手冊

## 系統元件

▼表 11-1‑1：系統之元件資訊

|  |  |
| --- | --- |
| 元件名稱 | MEI食不打烊 |
| 費用 | 免費 |
| 版本需求 | iOS全系列、Android 8以上 |
| 網路需求 | Wifi / 4G以上網路 |
| 支援語言 | 繁體中文 |

# 第12章　使用手冊

|  |  |
| --- | --- |
| 註冊頁面 | |
| ▲圖 11-1‑1：註冊頁面 | 首次登入的使用者尚未有帳號，需先註冊，輸入電子郵件、密碼及確認密碼後，按下註冊，即可註冊成功。 |
| 登入頁面 | |
| 一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片  AI 產生的內容可能不正確。  ▲圖 11-1‑2：登入頁面 | 註冊完畢後，點擊【登入】，輸入帳號及密碼後按下登入，即可進入到網頁。 |

|  |  |
| --- | --- |
| 首頁頁面 | |
|  |  |
| 站點查詢 | |
|  |  |
| 熱門站即時資訊 | |
|  |  |
| 當月TOP 10店家 | |
|  |  |
|  | |
|  |  |
|  | |
|  |  |

# 第13章　感想

* 11146061 郭宥妍

在本次專題之前，我曾在高中參與過類似大學專題的模式，當時同樣擔任組長，負責統籌並帶領團隊進行系統的開發與製作，整體規模皆比照大學專題進行。這段經驗讓我在面對此次專題時更加從容，自信地面對各種流程與挑戰，不僅不會感到緊張，也能協助組員更快熟悉進度，讓整體製作過程更加順利與高效。與以往不同的是，這次我首次接觸UI/UX設計。在剛開始使用Figma進行介面設計時，由於系統介面全為英文，加上毫無使用經驗，讓我一開始完全不知從何著手。為了克服這個障礙，我積極查閱教學資源與範例，並透過不斷的練習與嘗試，逐漸熟悉操作方式，也對設計的流程與原則有了初步的理解，這是我過去未曾接觸的領域，帶給我全新的挑戰與收穫。

此外，我在團隊領導與溝通協調方面也有明顯成長。我學會了如何更有效地分配工作內容，妥善安排會議時間，並在與老師討論時清楚表達進度與問題，同時也能處理各種突發的瑣碎事項。這次專題不僅提升了我的技術與設計能力，也讓我在組織、溝通與領導方面更加成熟，為未來面對更大專案奠定了穩固的基礎。

* 11146066 簡孝宇

這次專題從「把資料串起來」開始，一路走到能用的服務，心情像坐捷運轉好幾次車。最難的是資料對不起來、連線有時失敗、手機畫面又常走鐘；常常改好一處，另一處又壞。後來把目標拆小：先讓基本查詢能用，再慢慢補照片、路線、導航。遇到怪問題就先記錄步驟、重現一次、能用最簡單的方法先通過，之後再美化。過程中學到耐心與溝通的重要：誰負責哪一塊、改了要說清楚、寫下備忘與說明。雖然不完美，但看見頁面一步步成形，很有成就感；之後也想補更多提示、提升速度，讓整體更順更友善。

* 11146073 張楷偉

在這次的專題製作過程中，我學到了許多課堂上接觸不到的實作經驗。從一開始構思主題、規劃架構，到實際撰寫前端畫面與資料呈現的過程中，一路上也遇到很多的難關，例如在設計網頁版面時，要兼顧排版、美觀與響應式設計（RWD），常常一個小細節就得反覆修改；在處理資料與互動功能時，也要考慮使用者體驗與操作流程，這些都讓我更加了解網頁開發的細節與難處。

過程中也經歷過不少挫折，像是版面跑版、功能不如預期、檔案路徑錯誤等問題，常常需要花許多時間除錯。不過透過不斷嘗試、查資料與請教同學，逐漸把問題一一解決。看到網頁從草稿一步步變成可實際操作的成品，心中充滿成就感。這次專題不僅讓我提升了網頁設計的能力，也讓我學會面對困難時保持耐心、解決問題的態度，這將成為我未來學習與成長的重要養分。

* 11146083 林勝威

本次專題負責資料庫、後端、虛擬機的整合，在資料庫的部分，前期建置商家資料有遇到許多問題，像是要如何完整的取出商家的留言，原本想說使用google place api就可以將留言都抓下，後來發現是可以抓留言，但一次只有五筆的資料，後來將爬蟲技能用上來，才將大筆大筆的資料抓下，最後帶入到資料庫當中，後端在建置api時也有許多的困難，要先預想前端會需要那些資料，怎麼樣把資料傳遞會速度較快，當前端接收有問題時要幫忙解決哪裡出現了bug，虛擬機的部份為了要一直update餐廳頻論，所以有做了一些檔案讓虛擬機也能更新資料，本地端移到虛擬機上又會是新的考驗。

# 第14章　參考資料

1. ChatGPT
2. UI/UX Figma <https://www.youtube.com/@CreativeKev90>
3. 餐廳評論爬蟲<https://ithelp.ithome.com.tw/m/articles/10267131>
4. 圖片存放位置<https://ithelp.ithome.com.tw/articles/10330627?sc=pt>
5. 北捷api <https://www.metro.taipei/cp.aspx?n=BDEB860F2BE3E249>
6. Django <https://ithelp.ithome.com.tw/articles/10206355>
7. 批次檔定期更新 <https://ithelp.ithome.com.tw/articles/10369742>
8. Django資料庫 <https://ithelp.ithome.com.tw/m/articles/10301639>

# 附錄

評審問題回覆

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 宜補上定時撈取餐廳資料的介紹。 | 我們有在虛擬機設置一天一次抓取餐廳的資料，並且在每天凌晨3點都會更新。 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 檢索排序納入餐廳打烊時間、走路距離、人潮、留言的考量。 | 我們有新增餐廳從幾點營業到打烊時間，以及利用Google map計算從捷運站步行至餐廳的時間，排序方式為步行時間近到遠排序。 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 「美食打卡」可導入Google評論資料。 | 我們團隊目前技術尚未能達到此技術，且另一方面擔心導入Google評論後，會將我們原本系統上的評論給刷掉，因此我們採用外接方式導到Google評論。 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 結合代駕服務、深夜便利商店及喝酒專區，強化夜貓子功能。 | 我們有新增「喝酒專區」，此專區從夜貓子專區進入，喝酒專區內有酒吧和夜店的店家資訊及代駕專區，代駕專區我們直接連結到Uber的頁面，另外也有加上捷運末五班車的資訊在喝酒專區，可讓用戶不需再跳轉到上一頁搜尋捷運末班車。 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 站點搜尋不要用下拉式選單，可以用地圖呈現比較直觀。 | 我們考量到製作時間且下拉式選單較比直接地圖點選方式更直觀，所以我們決定維持原樣使用下拉式選單來搜尋站點。 |

|  |  |
| --- | --- |
| 評審建議事項 | 修正情形 |
| 未來可建議讓民眾導入行程，提供交通資訊和推薦美食功能，可參考MaaS融合食宿遊購行概念。 | 我們接受此建議，但目前尚未能提供這項服務，我們預期在複評後可以增進這項服務。 |