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114403組**

**題 目：欸！愛多益**

**指導老師：林宏仁老師**

**組 長：11146077林若庭**

**組 員：11046090李明隆 11146063范旖旎**

**11146071何玗柔 11146072林志恆**

**中華民國114年5月28日**

**目錄**

[第1章 前言 7](#_Toc209916225)

[1-1 背景介紹 7](#_Toc209916226)

[1-2 動機 7](#_Toc209916227)

[1-3 系統目的與目標 7](#_Toc209916228)

[1-4 預期成果 8](#_Toc209916229)

[第2章 營運計畫 10](#_Toc209916230)

[2-1 可行性分析 10](#_Toc209916231)

[2-2 商業模式－Business model 11](#_Toc209916232)

[2-3 市場分析－STP 11](#_Toc209916233)

[2-4 競爭力分析SWOT-TOWS 12](#_Toc209916234)

[第3章 系統規格 14](#_Toc209916235)

[3-1 系統架構 14](#_Toc209916236)

[3-2 系統軟、硬體需求與技術平台 15](#_Toc209916237)

[3-3 使用標準與工具 16](#_Toc209916238)

[第4章 專案時程與組織分工 17](#_Toc209916239)

[4-1 專案時程 17](#_Toc209916240)

[4-2 專案組織與分工 18](#_Toc209916241)

[4-3 上傳GitHub紀錄 19](#_Toc209916242)

[第5章 需求模型 20](#_Toc209916243)

[5-1 使用者需求 20](#_Toc209916244)

[5-2 使用個案圖(Use case diagram) 22](#_Toc209916245)

[5-3 使用個案描述 23](#_Toc209916246)

[5-4 分析類別圖(Analysis class diagram) 28](#_Toc209916247)

[第6章 設計模型 30](#_Toc209916248)

[6-1 循序圖(Sequential diagram) 30](#_Toc209916249)

[6-2 設計類別圖(Design class diagram) 41](#_Toc209916250)

[第7章 實作模型 42](#_Toc209916251)

[7-1 佈署圖(Deployment diagram)。 42](#_Toc209916252)

[7-2 套件圖(Package diagram)。 43](#_Toc209916253)

[7-3 元件圖(Component diagram)。 44](#_Toc209916254)

[7-4 狀態機(State machine)、時序圖(Timing diagram) 45](#_Toc209916255)

[第8章 資料庫設計 47](#_Toc209916256)

[8-1 資料庫關聯表 47](#_Toc209916257)

[8-2 表格及其Meta data。 50](#_Toc209916258)

[第9章 程式 60](#_Toc209916259)

[9-1 元件清單及其規格描述 60](#_Toc209916260)

[9-2 其他附屬之各種元件 86](#_Toc209916261)

[第10章 測試模型 89](#_Toc209916262)

[第11章 操作手冊 90](#_Toc209916263)

[第12章 使用手冊 91](#_Toc209916264)

[第13章 感想 92](#_Toc209916265)

[第14章 參考資料 93](#_Toc209916266)

**圖目錄**

圖 2-2‑1商業模式圖 11

圖 3-1‑1系統架構圖 14

圖 3-1‑2系統功能架構圖 14

圖 4-1‑1專案時程甘特圖 17

圖 4-3‑1上傳GitHub紀錄 19

圖 5-2‑1使用個案圖 22

圖 5-3‑1註冊之活動圖 23

圖 5-3‑2登入之活動圖 24

圖 5-3‑3忘記密碼之活動圖 24

圖 5-3‑4修改密碼之活動圖 25

圖 5-3‑5測驗之活動圖 25

圖 5-3‑6查看學習報告之活動圖 26

圖 5-3‑7管理員管理題庫之活動圖 26

圖 5-3‑8使用者編輯個人資料之活動圖 27

圖 5-4‑1分析類別圖 28

圖 5-4‑2使用者與基本測驗記錄之類別圖 28

圖 5-4‑3管理與問題回報之類別圖 29

圖 5-4‑4測驗之類別圖 29

圖 6-1‑1註冊之循序圖 30

圖 6-1‑2登入之循序圖 31

圖 6-1‑3忘記密碼之循序圖 32

圖 6-1‑4修改密碼之循序圖 33

圖 6-1‑5測驗之循序圖 34

圖 6-1‑6查看學習報告之循序圖 35

圖 6-1‑7管理員管理題庫之循序圖 36

圖 6-1‑8使用者編輯個人資料之循序圖 37

圖 6-1‑9領取單字卡之循序圖 38

圖 6-1‑10單字關聯地圖之循序圖 39

圖 6-1‑11選擇興趣之循序圖 40

圖 6-1‑12點數變動查詢之循序圖 40

圖 6-2‑1設計類別圖 41

圖 7-1‑1佈署圖 42

圖 7-2‑1套件圖 43

圖 7-3‑1元件圖 44

圖 7-4‑1狀態機 45

圖 7-4‑2時序圖 46

圖 8-1‑1資料庫關聯圖 47

**表目錄**

表 1-3‑1本系統與其他學習平台之比較 8

表 2-4‑1 SWOT分析 12

表 2-4‑2 TOWS策略 12

表 3-2‑1 系統軟體需求表 15

表 3-2‑2 系統硬體需求表 15

表 3-2‑3 技術平台表 15

表 3-3‑1 使用工具表 16

表 8-1‑1 資料庫關聯參考關係與限制(Constraints) 48

表 8-2‑1 users-使用者資料 50

表 8-2‑2 DailyVocabulary-每日單字資料 51

表 8-2‑3 Phrase -片語資料 51

表 8-2‑4 UserVocabularyRecord-使用者單字紀錄 52

表 8-2‑5 DailyTestRecord-每日測驗紀錄 52

表 8-2‑6 PointTransaction-點數交易紀錄 53

表 8-2‑7 ListeningMaterial -聽力素材 53

表 8-2‑8 ReadingPassage -閱讀文章 54

表 8-2‑9 Question -題目 55

表 8-2‑10 Exam -測驗 56

表 8-2‑11 ExamSession-測驗場次 57

表 8-2‑12 ExamResult-測驗結果 58

表 8-2‑13 ExamQuestion-測驗題目 58

表 8-2‑14 UserAnswer -使用者作答 59

表 9-1‑1 models-資料模型 60

表 9-1‑2 view-主頁與個人資料相關 61

表 9-1‑3 view-認證與帳戶管理相關 61

表 9-1‑4 view-測驗功能相關 61

表 9-1‑5 view-學習與紀錄相關 62

表 9-1‑6 view-輔助與工具相關 62

表 9-1‑7 templates-主頁與主要功能頁面 63

表 9-1‑8 templates-測驗功能相關頁面 63

表 9-1‑9 templates-使用者管理與認證 64

表 9-1‑10 templates-密碼重設流程 64

表 9-2‑1 套件-核心框架與伺服器 86

表 9-2‑2 套件-資料庫與資料處理 86

表 9-2‑3 套件- API 溝通與網路請求 87

表 9-2‑4 套件- 輔助與開發工具 88

表 9-2‑5 套件- 型別提示與程式碼品質 88

# 前言

## 背景介紹

隨著數位學習的興起，越來越多學習者選擇透過線上平台提升語言能力。多益（TOEIC）測驗作為國際標準的英語能力評估工具，廣泛應用於求職與學術認證。然而，傳統的學習方式往往缺乏個人化，且學習者難以有效追蹤進度與掌握弱點。因此，建置一個專門針對多益學習的線上平台，能夠結合個人化推薦、即時測驗與數據分析，將有效提升學習效率。

## 動機

目前市場上的多益學習資源多以傳統書籍、線上影片課程或固定題庫的形式呈現，這些方式雖然提供基本的學習內容，但缺乏互動性與個人化學習機制，導致學習者難以找到弱點並根據自身需求有效學習。此外，多數學習者在準備多益考試時，往往缺乏系統化的進度追蹤與錯題分析機制，使得學習過程變得較為低效。因此，本系統的開發目標是透過數據分析與智慧推薦機制，幫助學習者有計畫地準備多益考試，並透過即時反饋提升學習成效。

## 系統目的與目標

近年來，隨著 TOEIC（多益）測驗在國際間的普及，許多學習者希望透過線上學習平台來提升應試能力。然而，目前市面上的 TOEIC 學習系統多以單一方式提供學習內容，例如單純的單字記憶、模擬測驗，或是社群討論，缺乏針對個別使用者需求的個人化學習機制。

本系統的開發目標如下：

1. 提供 AI 個人化學習體驗：根據使用者學習狀況，推薦適合的練習題目與學習計畫，提高學習效率。
2. 結合模擬測驗與即時回饋：使用者可進行多益模擬測驗，系統即時提供成績分析與建議。
3. 支援學習提醒與進度追蹤：透過 LINE 通知或 Email 提醒，幫助使用者維持學習習慣。
4. 創造點數機制：利用點數機制，讓使用者可依自身興趣生成AI文章測驗，提高作答及使用意願。

表 1-3‑1本系統與其他學習平台之比較

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 功能 | 本系統 | Duolingo | Memrise | OurScool |
| 學習內容 | 針對TOEIC設計，個性化推薦內容 | 多語言選擇，偏向日常語言學習 | 主要為單字記憶 | 多語言選擇，不同主題授課 |
| 模擬測驗 | 與正式TOEIC相近方式進行測驗 | 單元制，無完整測驗 | 以單字測驗為主 | 單字、文法選擇題 |
| 即時回饋 | 錯誤分析與學習建議 | 即時提供答案 | 提供學習建議 | 即時提供解答與說明 |
| 學習提醒 | LINE / Email 通知 | APP內部提醒 | APP內提醒 | 官方Line通知 |
| 適用對象 | TOEIC應試者 | 語言學習者 | 加強單字學習者 | 語言學習者 |
| 成本 | 免費/付費買點數 | 免費/付費高級版 | 免費/付費高級版 | 簽約式，每月繳費 |

## 預期成果

透過本系統的開發與實際應用，預期能夠幫助學習者提升多益應試能力，並透過個人化學習機制提高學習效率。使用者將能夠透過系統的學習建議與測驗回饋，掌握自身的學習進度，並針對弱點進行強化練習。此外，透過數據分析與視覺化的學習紀錄，學習者將能夠更有計畫地準備考試，提升學習的主動性與積極性。整體而言，本系統將為學習者提供一個高效、智慧化且個人化的多益學習環境，幫助他們更輕鬆地達成目標。

# 營運計畫

## 可行性分析

本系統的開發與推廣需要考量技術、經濟、操作及市場可行性，以確保能夠成功運行並達到預期目標。

技術可行性：

1. 本系統使用Django作為後端框架，確保系統穩定性與可擴展性。
2. 前端採用HTML、CSS、JavaScript，結合AI技術 提供個人化學習體驗。
3. 數據分析與測驗管理依靠機器學習演算法，優化學習推薦。

經濟可行性：

1. 開發成本主要為技術人員薪資、伺服器運行成本及行銷支出。
2. 採用初期免費，降低使用者進入門檻，功能升級（例：AI生成文章）則需要花費購買點數。
3. 透過企業合作（B2B）及廣告收益獲取另一收入。

操作可行性：

1. 系統採用直覺化UI設計，確保學習者能輕鬆操作。
2. 支援RWD設計，提高使用便利性。
3. 整合LINE / Email通知，提醒使用者學習進度。

市場可行性：

1. TOEIC為全球性英語考試，市場需求穩定且持續增長。
2. 針對在職人士、大學生、考試準備者提供個人化學習方案。
3. 透過數據分析，精準推送學習內容，提高用戶黏著度。

## 商業模式－Business model

圖 2-2‑1商業模式圖

## 市場分析－STP

1. 市場區隔（Segmentation）

依據學習需求與目標：將市場區分為學生、上班族、補習班學員。

依據學習方式：區分為自學者、輔導班學習者、企業內訓學習者。

1. 目標市場（Targeting）

主要目標族群：大學生、求職者、上班族，準備TOEIC考試提升競爭力。

次要目標族群：學校、補習班、企業，提供TOEIC訓練方案。

1. 市場定位（Positioning）

差異化優勢：以AI個人化推薦+TOEIC專業訓練+互動式學習機制，區隔於傳統學習平台（如 Duolingo、Memrise）。

品牌定位：「最適合應試者的AI TOEIC學習平台」，讓考生能夠快速提升TOEIC成績。

## 競爭力分析SWOT-TOWS

表 2-4‑1 SWOT分析

|  |  |
| --- | --- |
| 優勢Strengths   1. 針對TOEIC設計的專業內容 2. 模擬考試與即時反饋機制 | 劣勢Weaknesses   1. 需要大量測試數據來優化推薦系統 2. 初期品牌知名度較低 3. 需與現有競爭者競爭（如多鄰國） |
| 機會Opportunities   1. TOEIC市場需求穩定 2. 企業與學校有TOEIC教育需求 | 威脅Threats   1. 大型學習平台競爭激烈 2. 免費學習資源增多，影響付費意願 |

表 2-4‑2 TOWS策略

|  |  |
| --- | --- |
| **SO策略（利用優勢抓住機會）**   1. 透過AI智能學習機制，提供個人化學習建議，強化市場競爭力 2. 與TOEIC輔導班、學校合作，提供學習方案 | **WO策略（克服弱勢並抓住機會）**   1. 透過社群行銷（如YouTube、FB、IG）提高品牌知名度 2. 提供限量免費模擬測驗，再進一步吸引消費者花費購買文章 |
| **ST策略（利用優勢降低威脅）**   1. 持續更新TOEIC模擬試題，確保內容優勢 2. 強化學習回饋機制，提高用戶黏著度，降低流失率 | **WT策略（克服弱勢並降低威脅）**   1. 提供更高效的學習方法（如AI學習追蹤），對抗免費資源威脅 2. 持續優化使用者體驗，提升學習效率與吸引力 |

# 系統規格

## 系統架構

圖 3-1‑1系統架構圖

圖 3-1‑2系統功能架構圖

## 系統軟、硬體需求與技術平台

表 3-2‑1 系統軟體需求表

|  |  |
| --- | --- |
| **項目** | **說明 / 使用技術** |
| 作業系統 | 開發端：Windows 11 使用者端：Windows、macOS、Android、iOS |
| 開發工具 | Visual Studio Code |
| 程式語言 | Python 3.11（後端）、HTML / CSS / JS（前端） |
| Web 框架 | Django |
| 自動化流程工具 | n8n（流程設計與系統整合） |
| 資料庫系統 | MySQL 8.0 |
| API 介接方式 | RESTful API（使用 JSON 格式傳輸） |
| 網頁伺服器 | Django 內建伺服器 |

表 3-2‑2 系統硬體需求表

|  |  |
| --- | --- |
| **項目** | **說明 / 規格建議** |
| 使用者端設備需求 | 一般桌機、筆電、平板、手機可正常使用 |
| 伺服器端需求 | CPU：雙核心以上 RAM：4GB 以上 儲存空間：建議 50GB |

表 3-2‑3 技術平台表

|  |  |
| --- | --- |
| **項目** | **說明 / 使用工具** |
| 前端技術 | HTML、Tailwind CSS、JavaScript、Font Awesome |
| 版本管理 | 使用 GitHub |
| 整合工具 | n8n + Webhook + Django API + MySQL 資料庫整合 |

## 使用標準與工具

表 3-3‑1 使用工具表

|  |  |
| --- | --- |
| **開發輔助工具** | **工具名稱** |
| 程式碼編輯器 | Visual Studio Code |
| 版本控制工具 | Git, Github |
| 資料庫管理工具 | MySQL Client, HeidiSQL |
| API 測試工具 | Swagger |
| LLM 應用開發工具 | Langchain |
| 自動化流程工具 | n8n |
| LLM 模型 | OpenAI ChatGPT 4O |
| 製圖工具 | Medibang paint, Draw.io |
| 文件製作 | Microsoft 365 Word |
| 簡報製作 | Microsoft 365 PowerPoint, Canva |
| 溝通工具 | LINE, Email |

# 專案時程與組織分工

## 專案時程

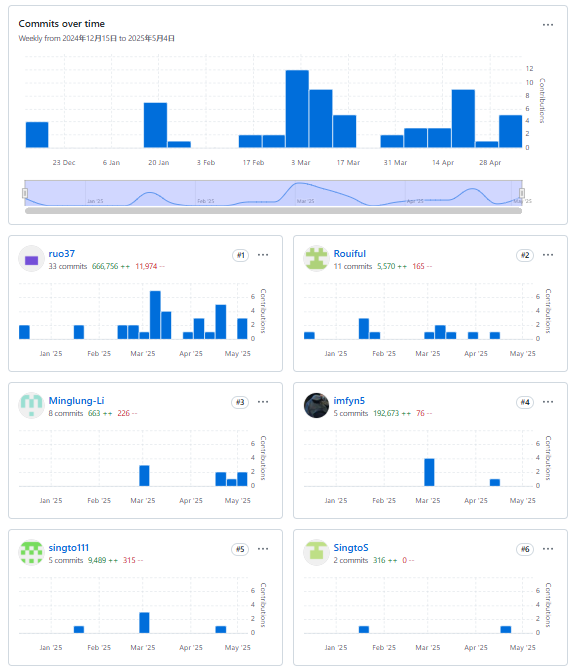
圖 4-1‑1專案時程甘特圖

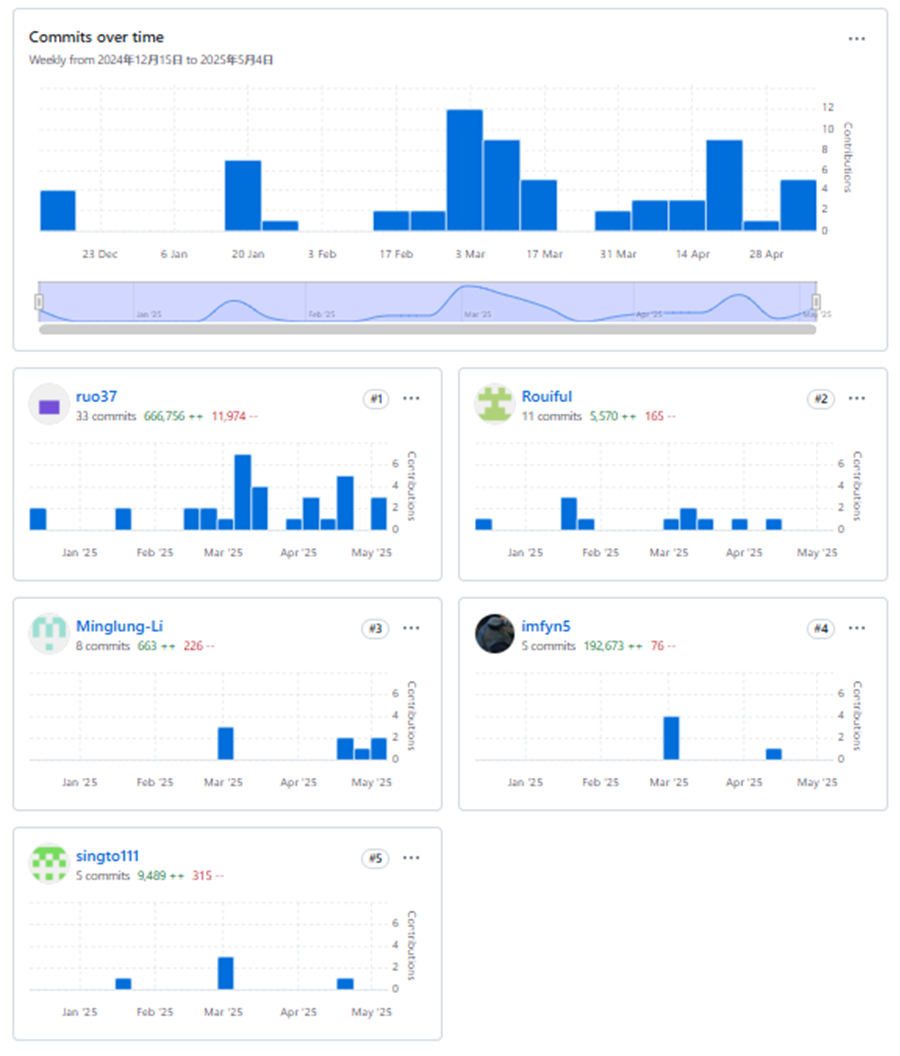
## 專案組織與分工

●主要負責人 〇次要負責人

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 項目/組員 | | 11146077林若庭 | 11046090李明隆 | 11146063范旖旎 | 11146071何玗柔 | 11146072林志恆 |
| 後端開發 | 資料庫建置 |  | ● | 〇 |  |  |
| 前後端整合 | ● |  | 〇 |  | 〇 |
| 帳號登入系統開發 |  |  | ● |  |  |
| 後台管理界面設計 |  |  |  |  | ● |
| 測驗平台功能設計 |  | 〇 | ● |  | 〇 |
| 測驗結果、學習歷程 | ● |  |  |  |  |
| 單字發送、單字關聯圖 | ● |  |  |  |  |
| API 設計與開發 |  | ● |  |  | 〇 |
| 前端開發 | 首頁、登入與註冊 |  |  | 〇 | ● |  |
| 測驗平台 |  |  |  | ● |  |
| 測驗結果、學習歷程 |  | ● |  | 〇 |  |
| 單字發送、單字關聯圖 | ● |  |  |  |  |
| 美術設計 | UI/ UX |  |  | 〇 | ● |  |
| Web介面設計 |  |  |  | ● |  |
| 色彩設計 |  |  | 〇 | ● |  |
| Logo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 |  |  | ● | 〇 |  |
| 第4章 專題時程與組織分工 | ● |  |  |  |  |
| 第5章 需求模型 | 〇 |  | 〇 |  | ● |
| 第6章 設計模型 |  | ● |  | 〇 |  |
| 第7章 實作模型 |  |  |  | 〇 | ● |
| 第8章 資料庫設計 | ● |  | 〇 |  |  |
| 第9章 程式 | ● |  | 〇 | 〇 |  |
| 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 |  |  |  | ● | 〇 |

## 上傳GitHub紀錄





11146071何玗柔

11146063范旖旎

11146072林志恆

11046090李明隆

11146077林若庭

圖 4-3‑1上傳GitHub紀錄

# 需求模型

## 使用者需求

5-1-1功能性需求

1. 使用者端
2. 帳號相關
   * 使用者可註冊新帳號。
   * 使用者可登入/登出帳號。
   * 使用者可查看與修改個人基本資料。
   * 使用者可變更密碼、申請忘記密碼功能。
3. 學習相關
   * 使用者可進行閱讀、聽力測驗。
   * 使用者可查詢學習歷程與分析報告。
   * 使用者每日可領取三張單字卡。
   * 使用者可查詢單字關聯地圖。
4. 點數相關
   * 使用者可使用點數進行測驗或兌換內容。
   * 使用者可查看點數變動情況。
5. 常見問題相關
   * 使用者可查看網站常見問題。
6. 管理者端
7. 使用者管理
   * 管理者可查詢／修改使用者資料。
   * 管理者可檢視使用者學習狀況與交易紀錄。
8. 題目與學習內容管理
   * 管理者可新增、修改、刪除題目。
   * 管理者可匯入／匯出題庫。
   * 管理者可設定題目分類與難易度。
9. 點數管理
   * 管理者可查詢點數交易紀錄。
   * 管理者可針對特定帳號進行點數調整（如取消、補發點數）。
   * 管理者可察看點數使用概況。
10. 常見問題管理
    * 管理者可維護常見問題（FAQ）。
11. 網站公告與內容管理
    * 管理者可新增與編輯首頁公告。
    * 管理者可發佈通知（如維護消息、活動資訊、更新通知）。

5-1-2非功能性需求

1. 裝置與網路支援
   * 系統應支援主流瀏覽器（Chrome、Firefox、Safari、Edge）。
   * 使用者可透過行動裝置或桌機使用系統。
   * 系統應支援有線網路、WiFi、行動網路（3G / 4G / 5G）。
2. 系統穩定性
   * 系統應能承載高併發使用者流量。
   * 所有學習資料應即時儲存、避免資料遺失。
3. 安全性
   * 使用者密碼應加密儲存。
   * 所有交易與個資傳輸需透過 HTTPS。
   * 管理者操作應有權限驗證與記錄。

## 使用個案圖(Use case diagram)

圖 5-2‑1使用個案圖

## 使用個案描述

圖 5-3‑1註冊之活動圖

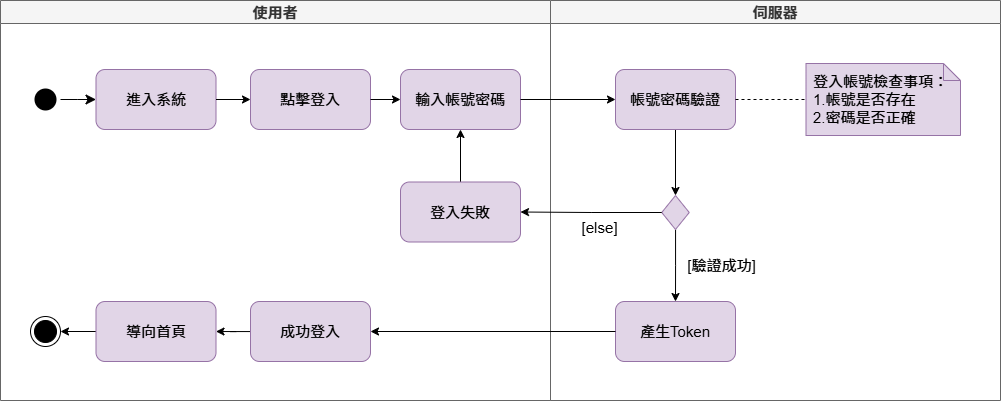
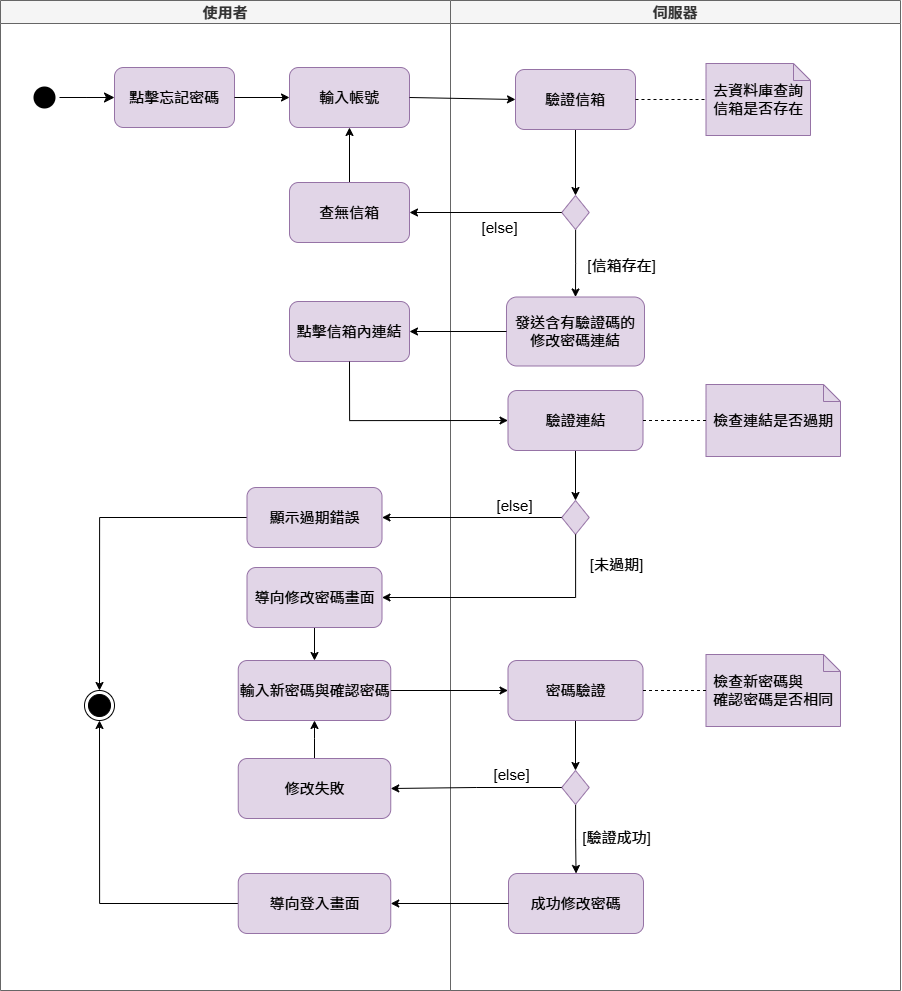
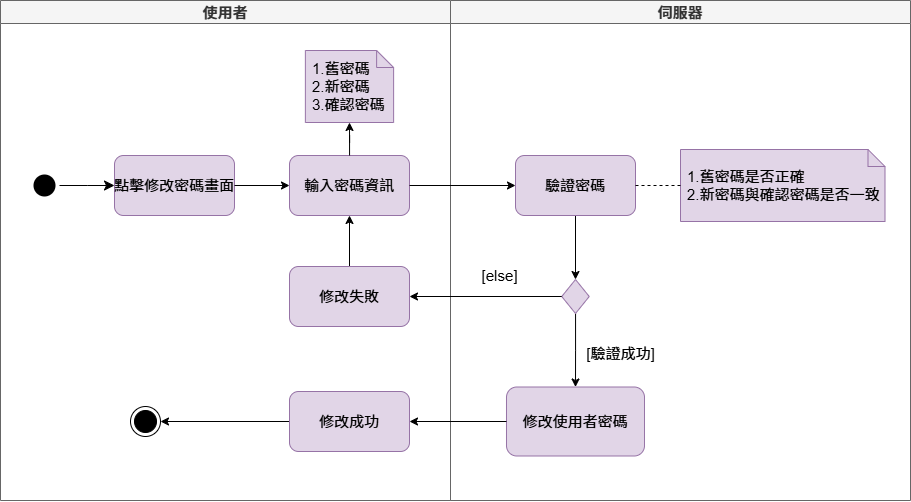
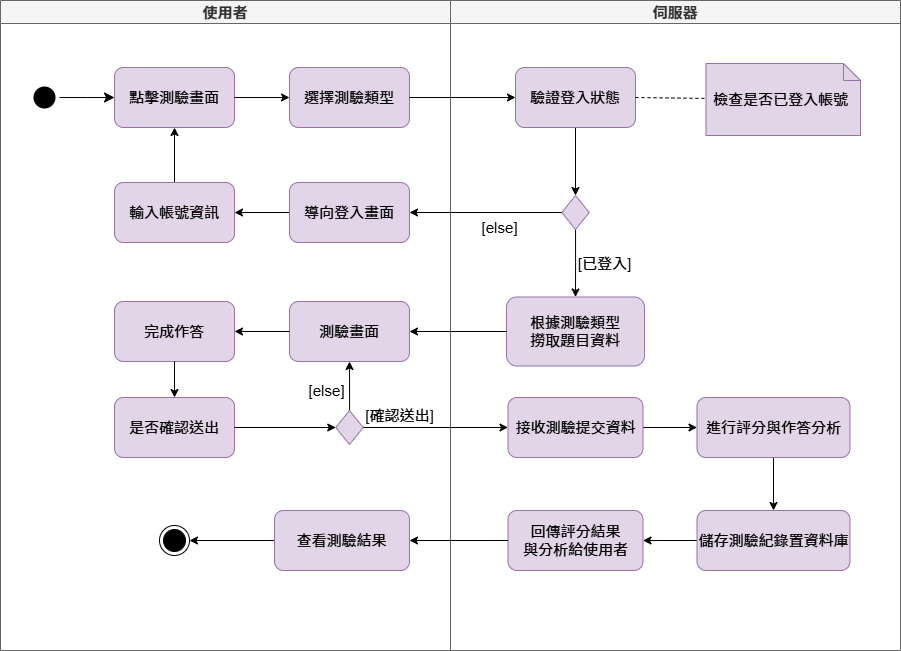
圖 5-3‑2登入之活動圖

圖 5-3‑3忘記密碼之活動圖

圖 5-3‑4修改密碼之活動圖

圖 5-3‑5測驗之活動圖

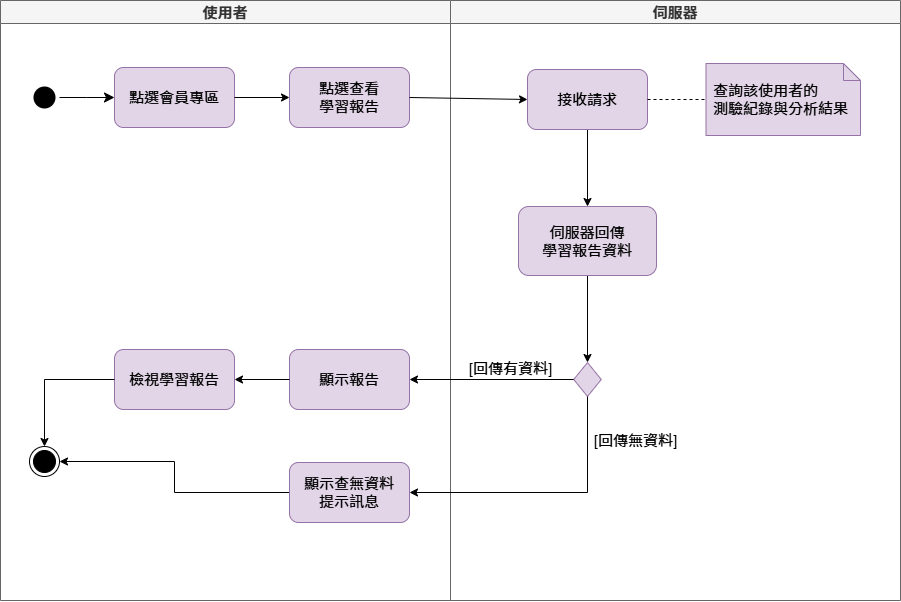
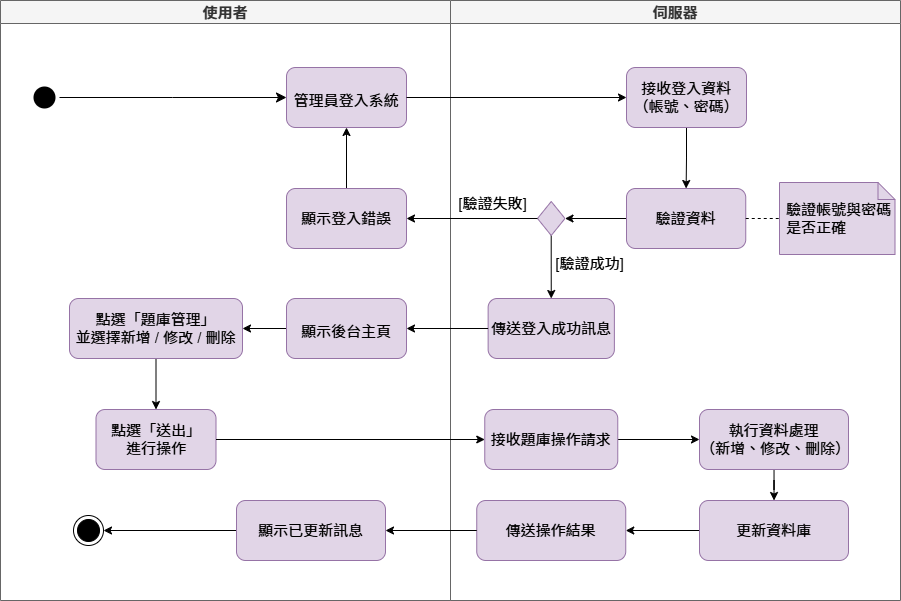
圖 5-3‑6查看學習報告之活動圖

圖 5-3‑7管理員管理題庫之活動圖

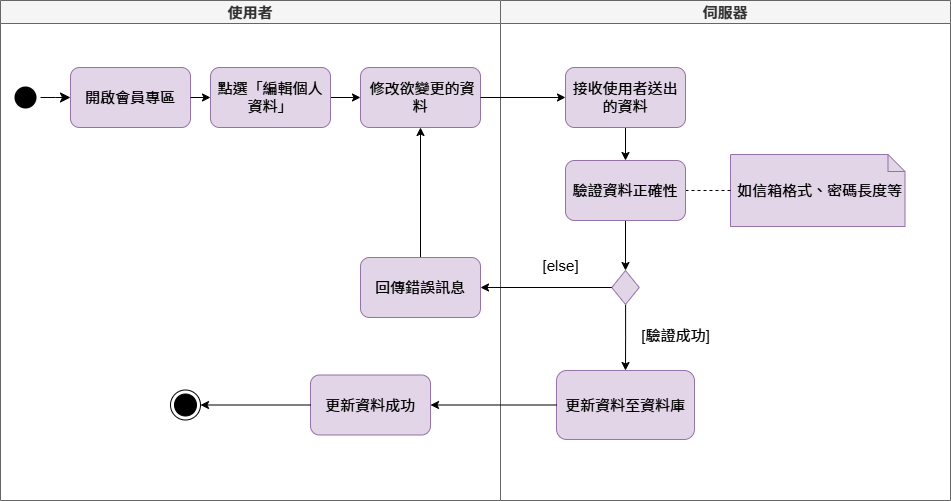


圖 5-3‑8使用者編輯個人資料之活動圖

## 分析類別圖(Analysis class diagram)

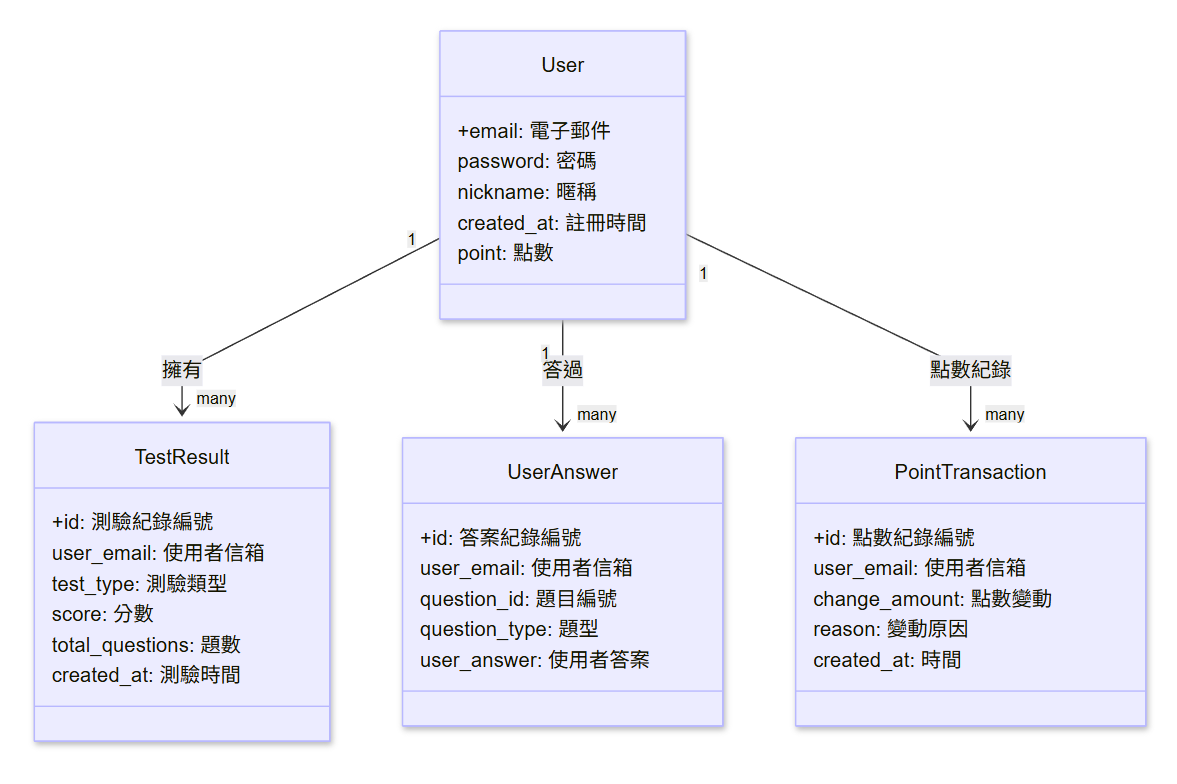
圖 5-4‑1分析類別圖

圖 5-4‑2使用者與基本測驗記錄之類別圖

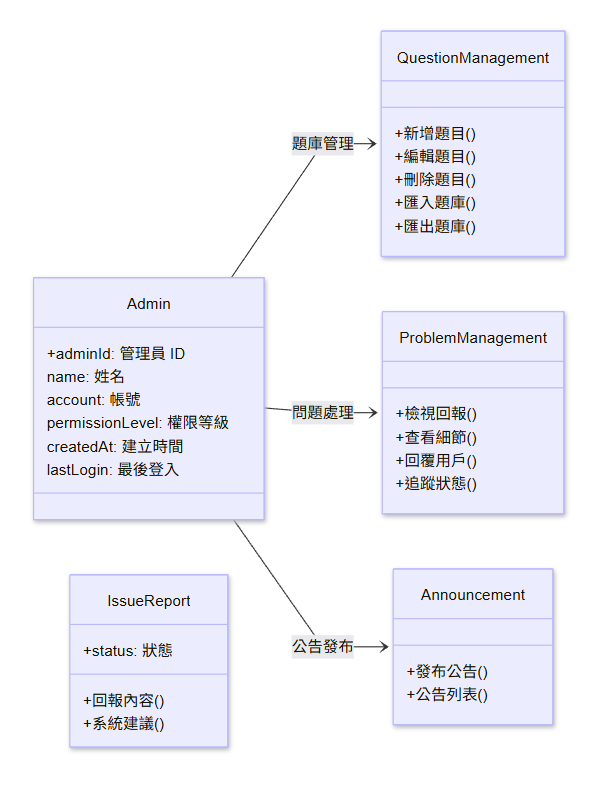
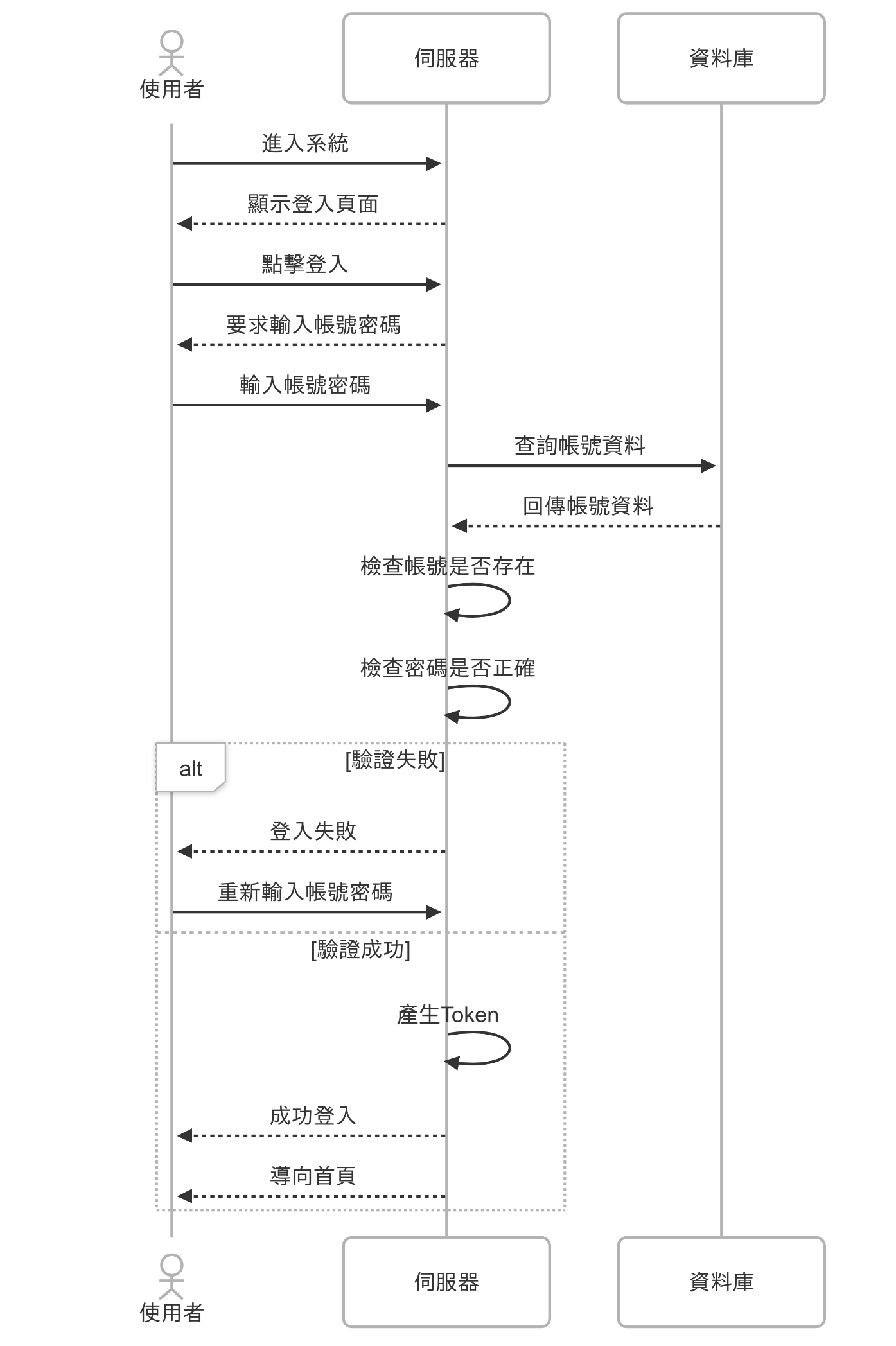
圖 5-4‑3管理與問題回報之類別圖

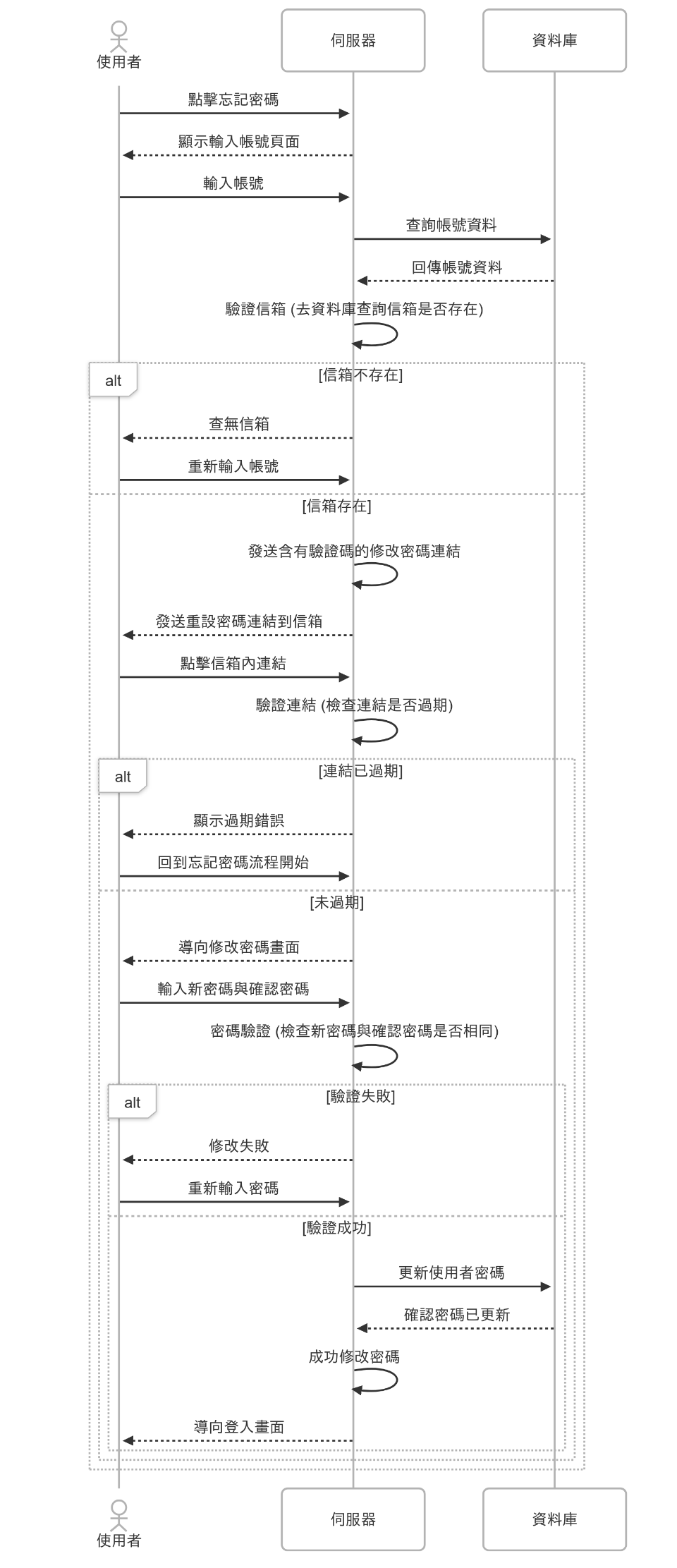
圖 5-4‑4測驗之類別圖

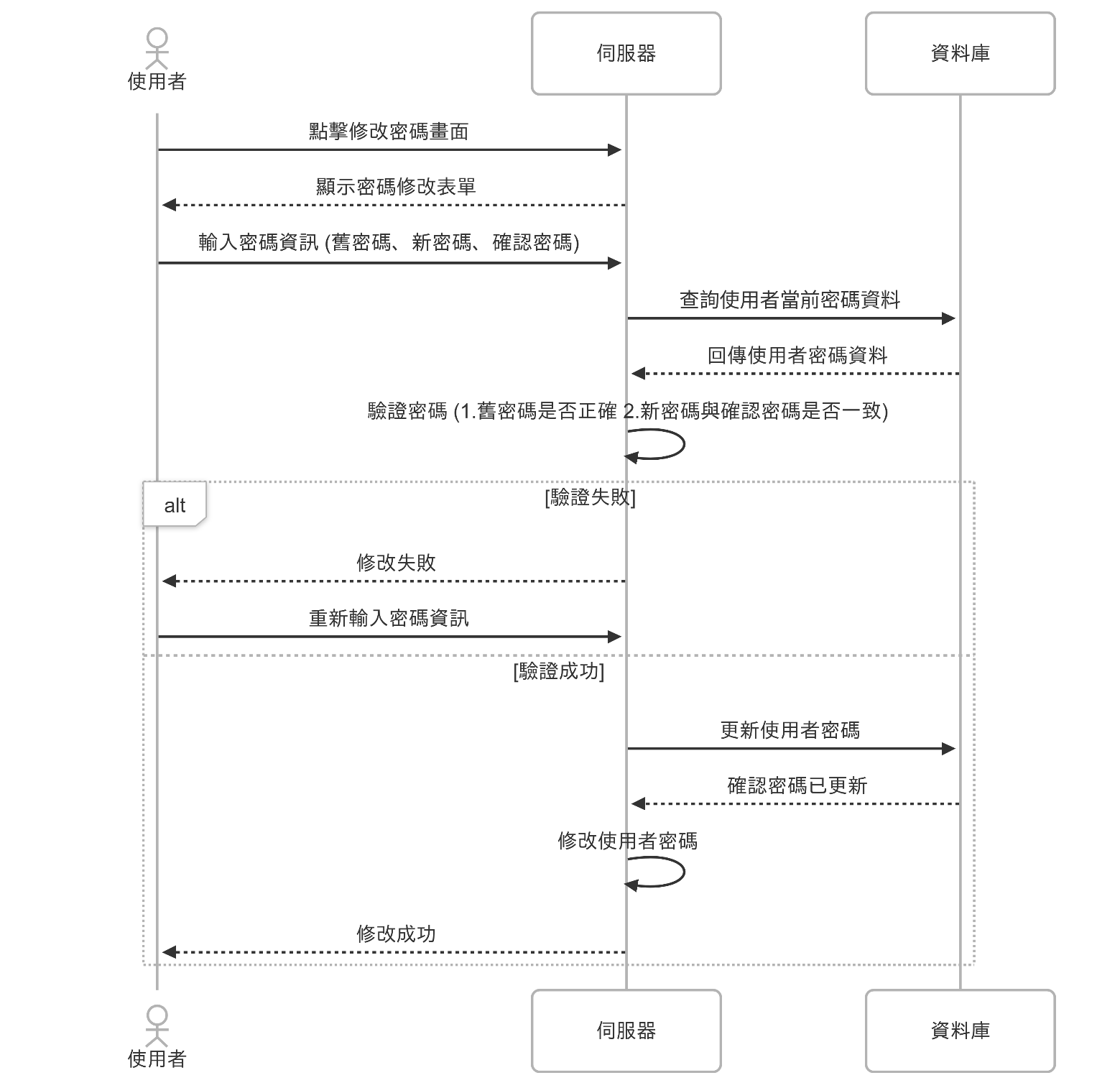
# 設計模型

## 循序圖(Sequential diagram)

圖 6-1‑1註冊之循序圖

圖 6-1‑2登入之循序圖

圖 6-1‑3忘記密碼之循序圖

圖 6-1‑4修改密碼之循序圖

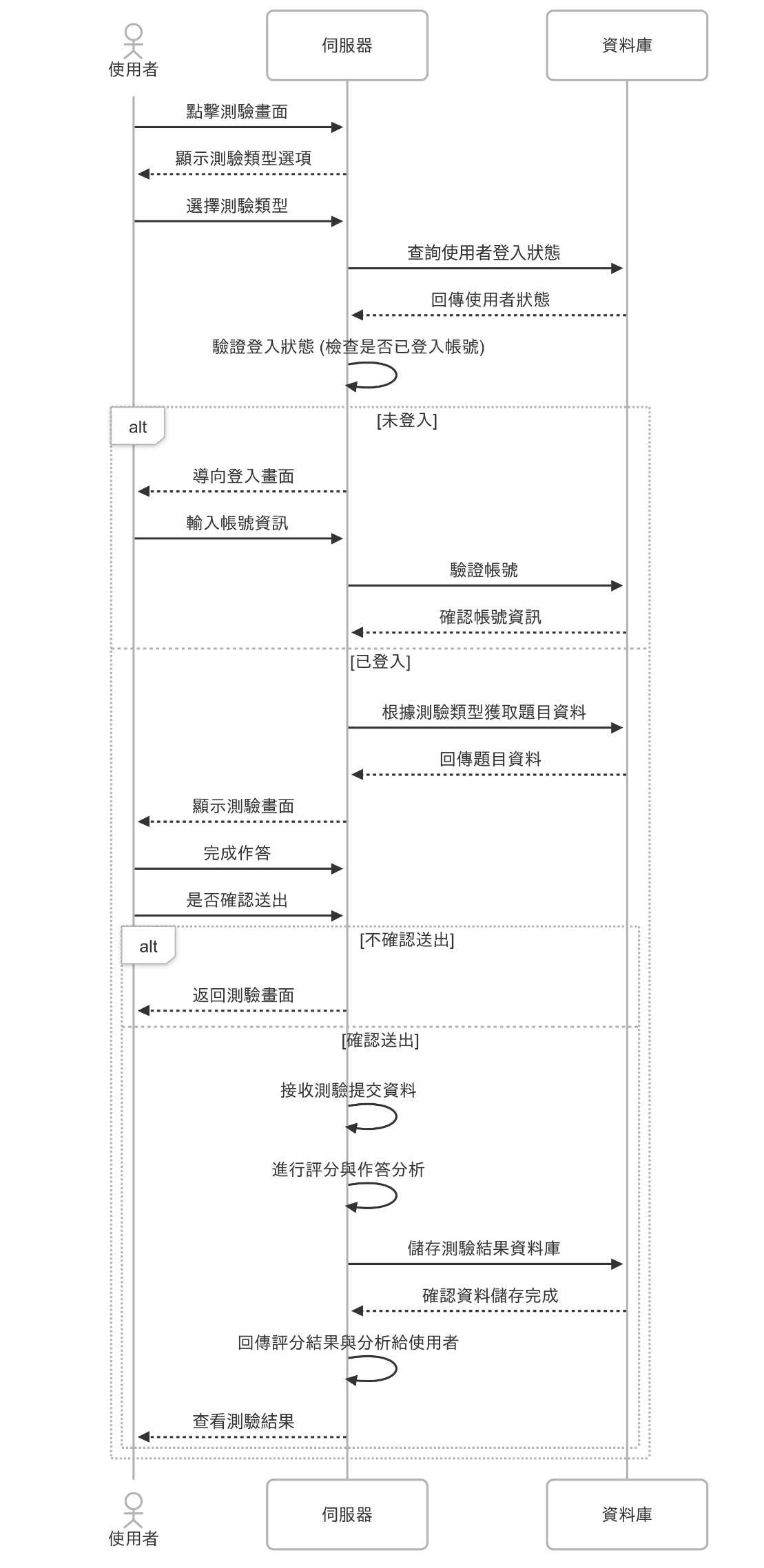
圖 6-1‑5測驗之循序圖

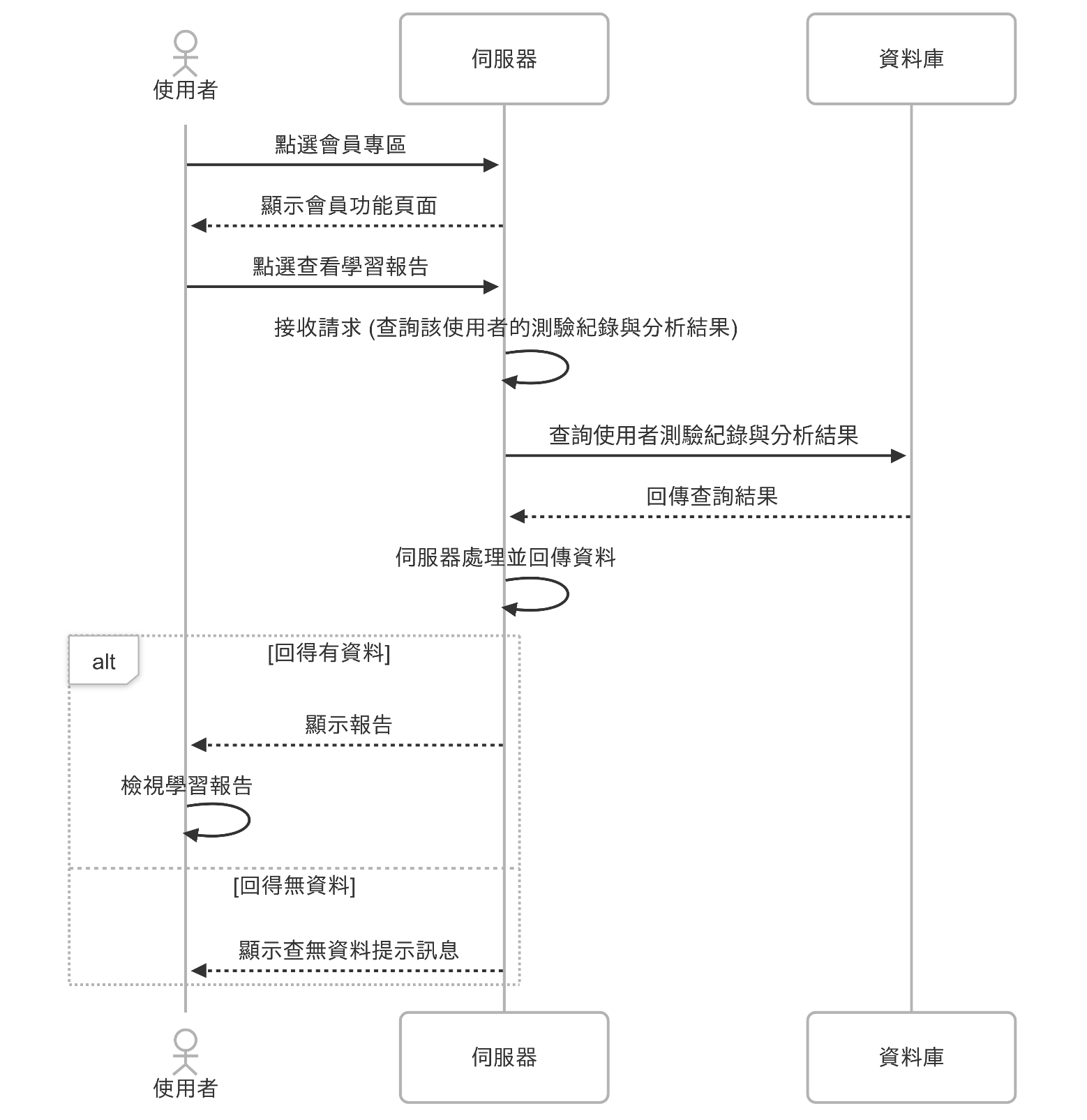
圖 6-1‑6查看學習報告之循序圖

圖 6-1‑7管理員管理題庫之循序圖

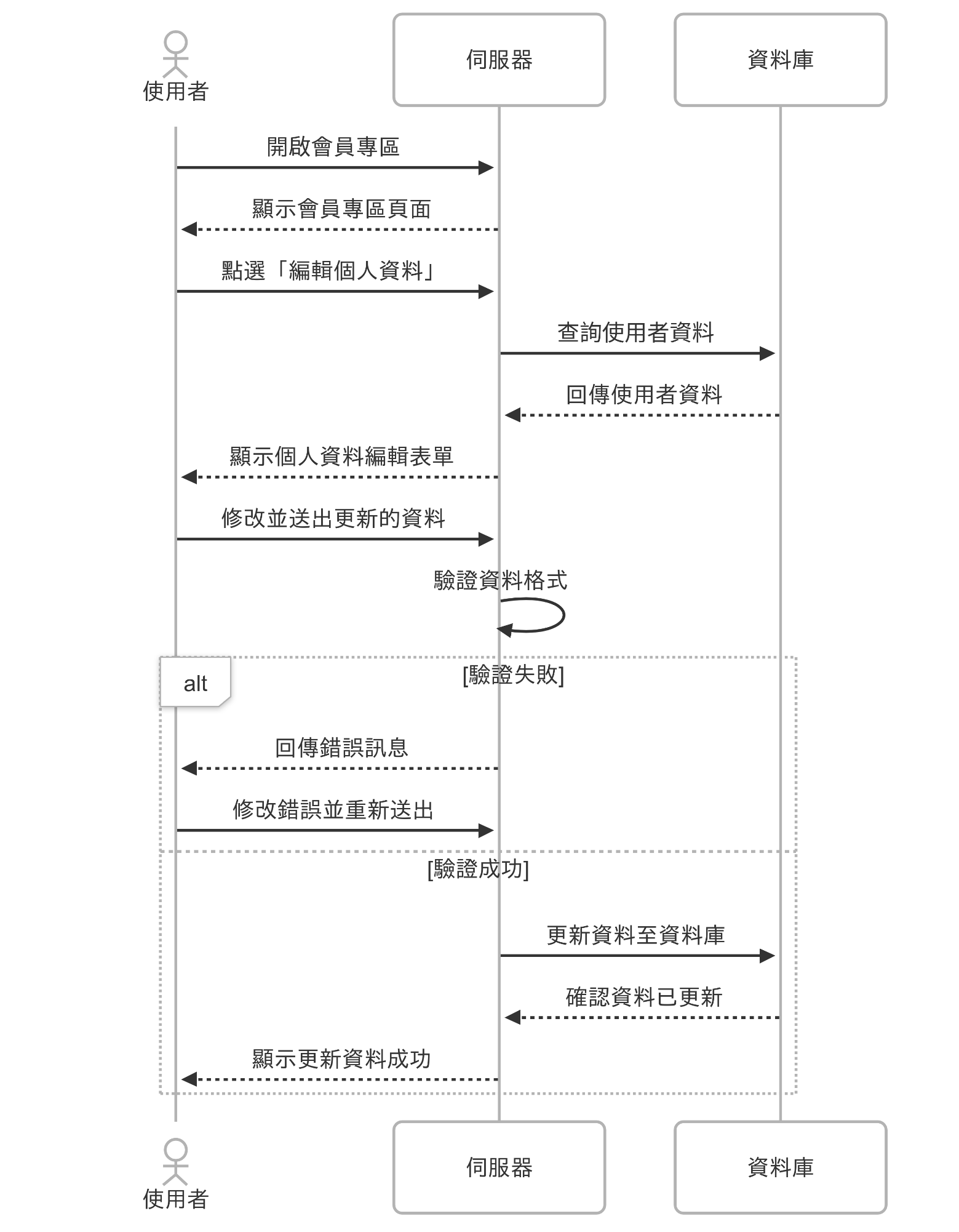


圖 6-1‑8使用者編輯個人資料之循序圖

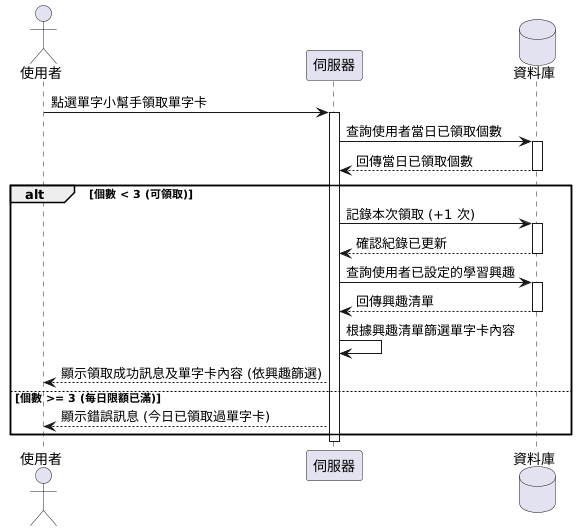


圖 6-1‑9領取單字卡之循序圖

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 6-1‑10單字關聯地圖之循序圖

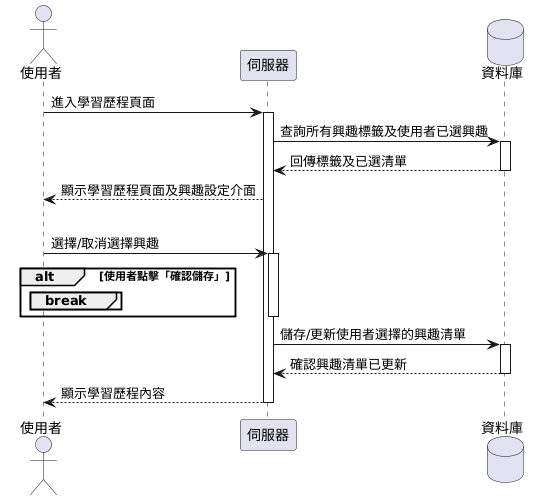


圖 6-1‑11選擇興趣之循序圖

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 6-1‑12點數變動查詢之循序圖

## 設計類別圖(Design class diagram)

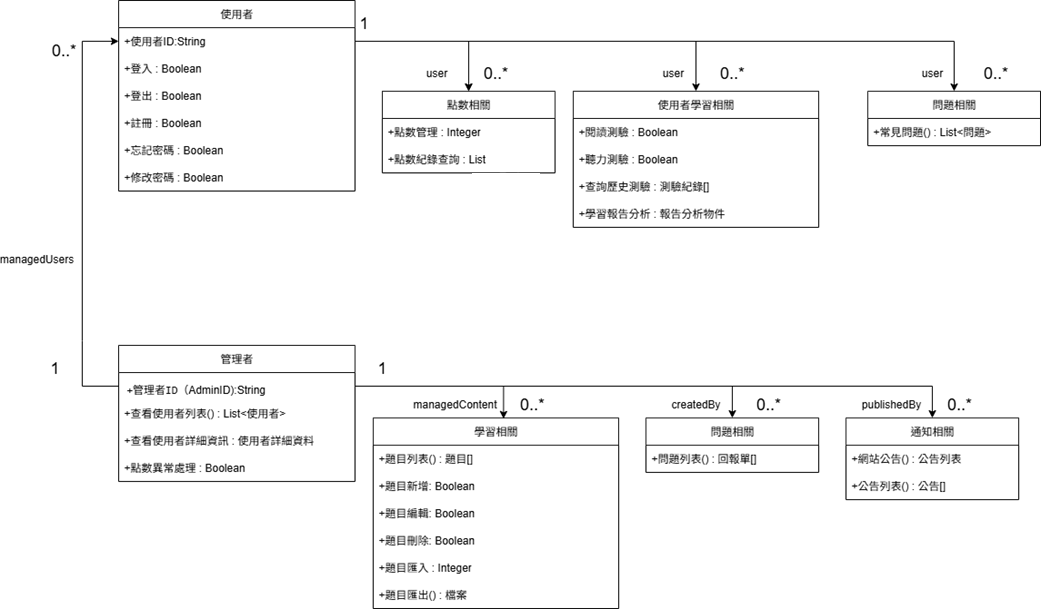


圖 6-2‑1設計類別圖

# 實作模型

## 佈署圖(Deployment diagram)。



圖 7-1‑1佈署圖

## 套件圖(Package diagram)。

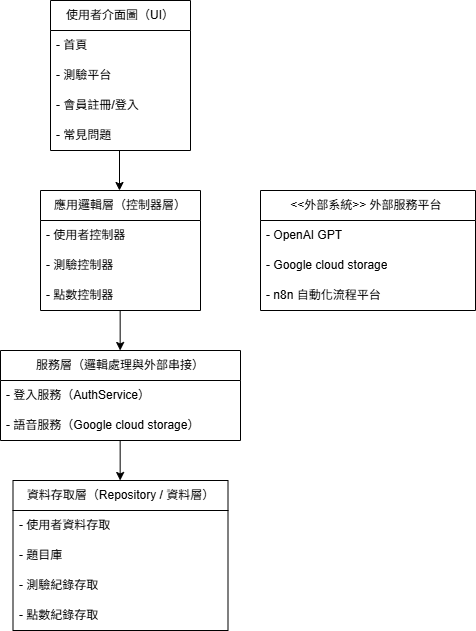


圖 7-2‑1套件圖

## 元件圖(Component diagram)。

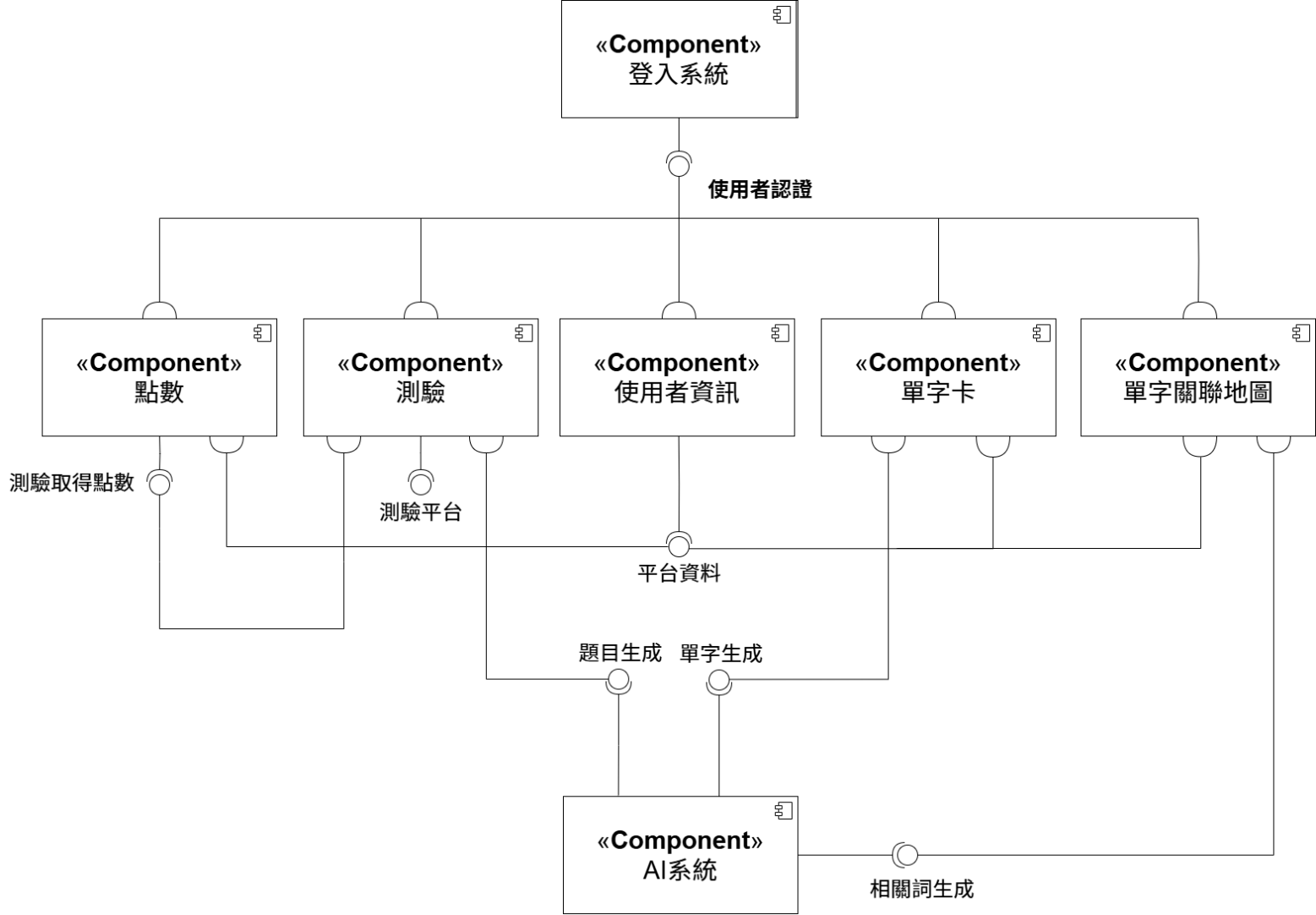


圖 7-3‑1元件圖

## 狀態機(State machine)、時序圖(Timing diagram)

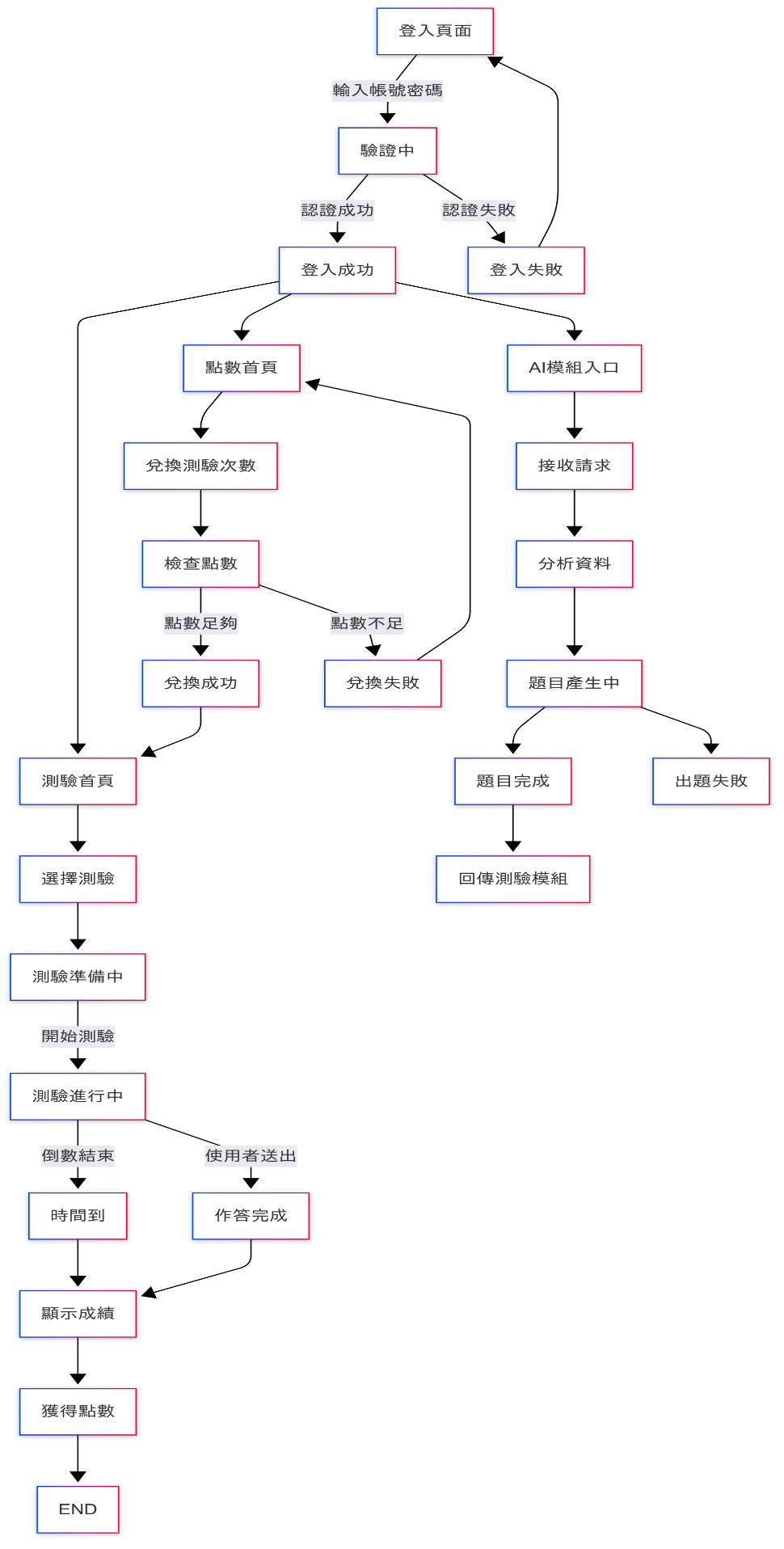


圖 7-4‑1狀態機

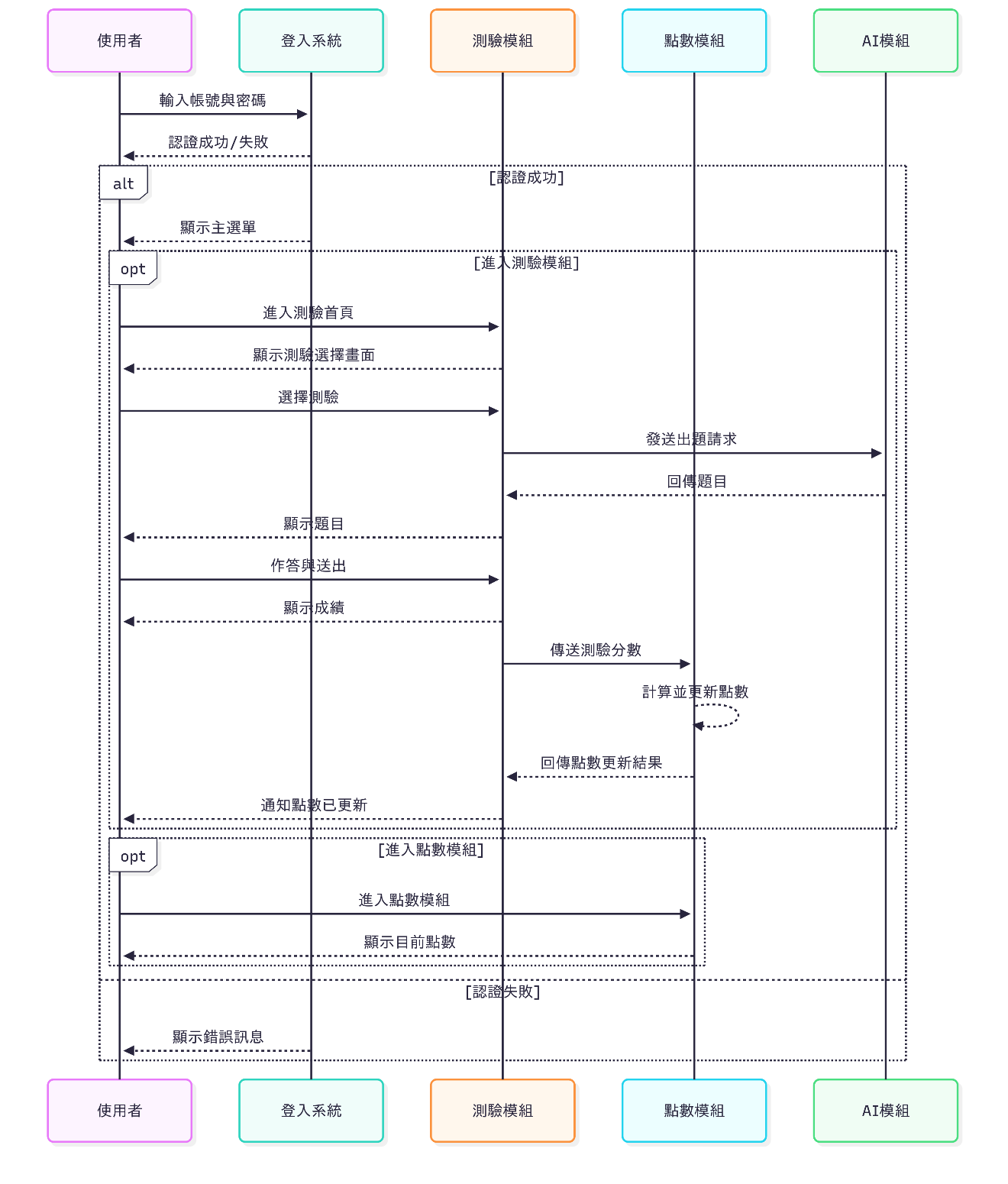


圖 7-4‑2時序圖

# 資料庫設計

## 資料庫關聯表

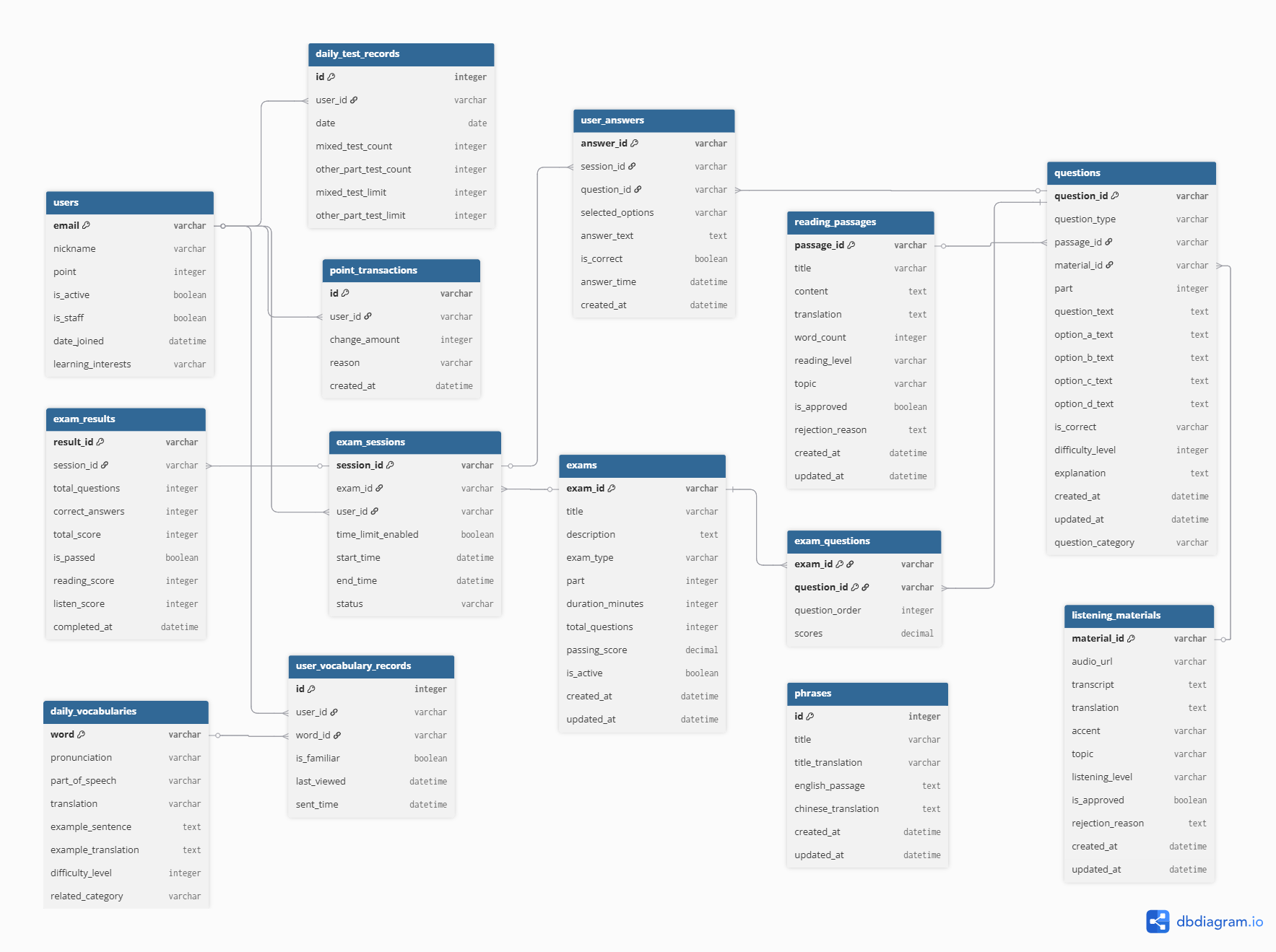


圖 8-1‑1資料庫關聯圖

表 8-1‑1 資料庫關聯參考關係與限制(Constraints)

|  |  |  |  |
| --- | --- | --- | --- |
| 資料表名稱 | 參考對象 | 關聯方式 | 限制條件  (Constraints) |
| user | 無 | - | email為 PRIMARY KEY，NOT NULL。 |
| DailyVocabulary | 無 | - | word 欄位為 UNIQUE。 |
| Phrase | 無 | - | id 為 PRIMARY KEY。 |
| UserVocabularyRecord | User, DailyVocabulary | user → User， word → DailyVocabulary | 外鍵，ON DELETE CASCADE；(user, word) 組合必須為 UNIQUE。 |
| DailyTestRecord | User | user → User | 外鍵，ON DELETE CASCADE；(user, date) 組合必須為 UNIQUE。 |
| PointTransaction | User | user → User | 外鍵，ON DELETE CASCADE；reason 為 ENUM。 |
| ListeningMaterial | 無 | - | material\_id 為 PRIMARY KEY；listening\_level, accent 為 ENUM。 |
| ReadingPassage | 無 | - | passage\_id 為 PRIMARY KEY；reading\_level, rejection\_reason 為 ENUM。 |
| Question | ReadingPassage, ListeningMaterial | passage → ReadingPassage， material → ListeningMaterial | 外鍵可為 NULL，ON DELETE SET NULL；question\_type, part, difficulty\_level, question\_category 皆為 ENUM。 |
| Exam | 無 | - | exam\_id 為 PRIMARY KEY；exam\_type, part 為 ENUM。 |
| ExamSession | Exam, User | exam → Exam，user → User | 外鍵，ON DELETE CASCADE；session\_id 為 PRIMARY KEY；status 為 ENUM。 |
| ExamResult | ExamSession | session → ExamSession (One-to-One) | 外鍵，ON DELETE CASCADE；result\_id 為 PRIMARY KEY。 |
| ExamQuestion | Exam, Question | exam → Exam，question → Question | 外鍵，ON DELETE CASCADE；(exam, question) 組合必須為 UNIQUE。 |
| UserAnswer | ExamSession, Question | session → ExamSession，question → Question | 外鍵，ON DELETE CASCADE；answer\_id 為 PRIMARY KEY。 |

## 表格及其Meta data。

表 8-2‑1 users-使用者資料

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| email | EmailField (VARCHAR) | 使用者信箱 | PRIMARY KEY, UNIQUE, NOT NULL |
| password | CharField (VARCHAR) | 加密密碼 | NOT NULL |
| nickname | CharField (VARCHAR) | 暱稱 | NOT NULL  (在REQUIRED\_FIELDS 中) |
| point | IntegerField | 點數餘額 | 預設值為 0 |
| is\_active | BooleanField | 帳號是否啟用 | 預設值為 True |
| is\_staff | BooleanField | 是否為後台管理員 | 預設值為 False |
| date\_joined | DateTimeField | 帳號建立時間 | auto\_now\_add=True |
| learning\_interests | CharField (VARCHAR) | 學習興趣 | 可為空字串，預設值為 '' |

表 8-2‑2 DailyVocabulary-每日單字資料

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| word | CharField (VARCHAR) | 英文單字 | UNIQUE, NOT NULL |
| pronunciation | CharField (VARCHAR) | 音標 | 可為空 |
| part\_of\_speech | CharField (VARCHAR) | 詞性 | NOT NULL |
| translation | CharField (VARCHAR) | 中文翻譯 | NOT NULL |
| example\_sentence | TextField | 例句 | NOT NULL |
| example\_translation | TextField | 例句翻譯 | 可為空或 NULL |
| difficulty\_level | IntegerField | 難度等級 | 預設值為 1 |
| related\_category | CharField (VARCHAR) | 主題類別 | 預設值為 'DailyLife', 必須為 CATEGORY\_CHOICES 中的選項 |

表 8-2‑3 Phrase -片語資料

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| id | AutoField (INTEGER) | 短語編號 | PRIMARY KEY |
| title | CharField (VARCHAR) | 標題 | NOT NULL |
| title\_translation | CharField (VARCHAR) | 標題中文翻譯 | 可為空或 NULL |
| english\_passage | TextField | 英文文章段落 | NOT NULL |
| chinese\_translation | TextField | 中文翻譯 | NOT NULL |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |
| updated\_at | DateTimeField | 更新時間 | auto\_now=True |

表 8-2‑4 UserVocabularyRecord-使用者單字紀錄

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| user | ForeignKey (VARCHAR) | 關聯的使用者 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| word | ForeignKey (VARCHAR) | 關聯的單字 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| is\_familiar | BooleanField | 是否熟悉該單字 | 預設值為 False |
| last\_viewed | DateTimeField | 最後檢視時間 | auto\_now=True |
| sent\_time | DateTimeField | 單字發送時間 | auto\_now\_add=True, 可為 NULL |

表 8-2‑5 DailyTestRecord-每日測驗紀錄

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| user | ForeignKey (VARCHAR) | 關聯的使用者 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| date | DateField | 測驗日期 | 預設值為今天日期 |
| mixed\_test\_count | IntegerField | 綜合測驗次數 | 預設值為 0 |
| other\_part\_test\_count | IntegerField | 單一 Part 測驗次數 | 預設值為 0 |
| mixed\_test\_limit | IntegerField | 綜合測驗次數限制 | 預設值為 1 |
| other\_part\_test\_limit | IntegerField | 單一 Part 測驗次數限制 | 預設值為 3 |

表 8-2‑6 PointTransaction-點數交易紀錄

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| id | UUIDField | 交易 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| user | ForeignKey (VARCHAR) | 關聯的使用者 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| change\_amount | IntegerField | 點數變動量 | NOT NULL |
| reason | CharField (VARCHAR) | 變動原因 | NOT NULL，必須為 POINT\_REASON\_CHOICES 中的選項 |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |

表 8-2‑7 ListeningMaterial -聽力素材

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| material\_id | UUIDField | 聽力素材 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| audio\_url | CharField (VARCHAR) | 音檔 URL | 可為空或 NULL |
| transcript | TextField | 聽力逐字稿 | NOT NULL |
| translation | TextField | 逐字稿中文翻譯 | 可為空或 NULL |
| accent | CharField (VARCHAR) | 口音 | 可為空或 NULL，必須為 ACCENT\_CHOICES 中的選項 |
| topic | CharField (VARCHAR) | 主題 | NOT NULL |
| listening\_level | CharField (VARCHAR) | 難度等級 | NOT NULL，必須為 LISTENING\_LEVEL\_CHOICES 中的選項 |
| is\_approved | BooleanField | 是否已審核通過 | 預設值為 False |
| rejection\_reason | TextField | 拒絕原因 | 可為空或 NULL |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |
| updated\_at | DateTimeField | 更新時間 | auto\_now=True |

表 8-2‑8 ReadingPassage -閱讀文章

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| passage\_id | UUIDField | 文章 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| title | CharField (VARCHAR) | 標題 | NOT NULL |
| content | TextField | 文章內容 | NOT NULL |
| translation | TextField | 文章中文翻譯 | 可為空或 NULL |
| word\_count | IntegerField | 字數 | NOT NULL |
| reading\_level | CharField (VARCHAR) | 難度等級 | NOT NULL，必須為 READING\_LEVEL\_CHOICES 中的選項 |
| topic | CharField (VARCHAR) | 主題 | NOT NULL |
| is\_approved | BooleanField | 是否已審核通過 | 預設值為 False |
| rejection\_reason | CharField (VARCHAR) | 拒絕原因 | 可為空或 NULL，必須為 REJECTION\_REASON\_CHOICES 中的選項 |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |
| updated\_at | DateTimeField | 更新時間 | auto\_now=True |

表 8-2‑9 Question -題目

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| question\_id | UUIDField | 題目 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| question\_type | CharField (VARCHAR) | 題目類型 | NOT NULL，必須為 QUESTION\_TYPE\_CHOICES 中的選項 |
| passage | ForeignKey (UUID) | 關聯的閱讀文章 | 外部鍵，可為 NULL，ON DELETE SET NULL |
| material | ForeignKey (UUID) | 關聯的聽力素材 | 外部鍵，可為 NULL，ON DELETE SET NULL |
| part | IntegerField | 測驗部分 | 可為空或 NULL，必須為 PART\_CHOICES 中的選項 |
| question\_text | TextField | 題目內文 | NOT NULL |
| option\_a\_text | TextField | 選項 A | NOT NULL |
| option\_b\_text | TextField | 選項 B | NOT NULL |
| option\_c\_text | TextField | 選項 C | NOT NULL |
| option\_d\_text | TextField | 選項 D | NOT NULL |
| is\_correct | CharField (VARCHAR) | 正確選項 | NOT NULL |
| difficulty\_level | IntegerField | 難度等級 | NOT NULL，必須為 DIFFICULTY\_LEVEL\_CHOICES 中的選項 |
| explanation | TextField | 解析 | NOT NULL |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |
| updated\_at | DateTimeField | 更新時間 | auto\_now=True |
| question\_category | CharField (VARCHAR) | 分類題目類型 | 預設值為 'vocab', 必須為 QUESTION\_CATEGORY\_CHOICES 中的選項 |

表 8-2‑10 Exam -測驗

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| exam\_id | UUIDField | 測驗 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| title | CharField (VARCHAR) | 標題 | NOT NULL |
| description | TextField | 說明 | NOT NULL |
| exam\_type | CharField (VARCHAR) | 測驗類型 | NOT NULL，必須為 EXAM\_TYPE\_CHOICES 中的選項 |
| part | IntegerField | 測驗部分 | 可為空或 NULL，必須為 PART\_CHOICES 中的選項 |
| duration\_minutes | IntegerField | 測驗時長(分鐘) | NOT NULL |
| total\_questions | IntegerField | 總題數 | NOT NULL |
| passing\_score | DecimalField | 及格分數 | NOT NULL |
| is\_active | BooleanField | 是否啟用 | 預設值為 True |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |
| updated\_at | DateTimeField | 更新時間 | auto\_now=True |

表 8-2‑11 ExamSession-測驗場次

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| session\_id | UUIDField | 會話 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| exam | ForeignKey (UUID) | 關聯的測驗 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| user | ForeignKey (VARCHAR) | 關聯的使用者 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| time\_limit\_enabled | BooleanField | 是否啟用時間限制 | 預設值為 True |
| start\_time | DateTimeField | 開始時間 | NOT NULL |
| end\_time | DateTimeField | 結束時間 | NOT NULL |
| status | CharField (VARCHAR) | 會話狀態 | NOT NULL，必須為 SESSION\_STATUS\_CHOICES 中的選項 |

表 8-2‑12 ExamResult-測驗結果

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| result\_id | UUIDField | 結果 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| session | OneToOneField (UUID) | 關聯的測驗會話 | 外部鍵，UNIQUE，NOT NULL，ON DELETE CASCADE |
| total\_questions | IntegerField | 總題數 | NOT NULL |
| correct\_answers | IntegerField | 答對題數 | NOT NULL |
| total\_score | IntegerField | 總分數 | NOT NULL |
| is\_passed | BooleanField | 是否通過 | NOT NULL |
| reading\_score | IntegerField | 閱讀分數 | NOT NULL |
| listen\_score | IntegerField | 聽力分數 | NOT NULL |
| completed\_at | DateTimeField | 完成時間 | NOT NULL |

表 8-2‑13 ExamQuestion-測驗題目

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| exam | ForeignKey (UUID) | 關聯的測驗 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| question | ForeignKey (UUID) | 關聯的題目 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| question\_order | IntegerField | 題目順序 | NOT NULL |
| scores | DecimalField | 該題分數 | NOT NULL |

表 8-2‑14 UserAnswer -使用者作答

|  |  |  |  |
| --- | --- | --- | --- |
| 欄位名稱 | 資料型態 | 說明 | 限制條件 |
| answer\_id | UUIDField | 作答 ID | PRIMARY KEY, UNIQUE, NOT NULL |
| session | ForeignKey (UUID) | 關聯的測驗會話 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| question | ForeignKey (UUID) | 關聯的題目 | 外部鍵，NOT NULL，ON DELETE CASCADE |
| selected\_options | CharField (VARCHAR) | 使用者選擇的選項 | NOT NULL |
| answer\_text | TextField | 使用者作答內容 | 可為空或 NULL |
| is\_correct | BooleanField | 是否答對 | NOT NULL |
| answer\_time | DateTimeField | 作答時間 | NOT NULL |
| created\_at | DateTimeField | 建立時間 | auto\_now\_add=True |

# 程式

## 元件清單及其規格描述

表 9-1‑1 models-資料模型

|  |  |
| --- | --- |
| 資料表名稱 | 說明 |
| **User** | 使用者基本資料（Email 為主鍵，含暱稱、積分、權限等） |
| **DailyVocabulary** | 每日單字表，存放單字、詞性、翻譯與例句 |
| **Phrase** | 片語，含英文原文與中文翻譯 |
| **UserVocabularyRecord** | 使用者單字學習紀錄（是否熟悉、最後檢視時間） |
| **DailyTestRecord** | 使用者每日測驗次數與限制紀錄 |
| **PointTransaction** | 積分變動紀錄（完成測驗、兌換等） |
| **ListeningMaterial** | 聽力素材（音檔、逐字稿、口音、主題、審核狀態） |
| **ReadingPassage** | 閱讀文章（內文、翻譯、字數、主題、審核狀態） |
| **Question** | 題目資料（題目內容、選項、答案、解析、分類） |
| **Exam** | 測驗資訊（標題、類型、Part、時限、題數、及格分數） |
| **ExamSession** | 測驗場次（與使用者/測驗綁定，狀態追蹤） |
| **ExamResult** | 測驗結果（答題數、正確數、分數、是否通過） |
| **ExamQuestion** | 測驗與題目的對應表（含題目順序與分數） |
| **UserAnswer** | 使用者作答紀錄（選項、是否正確、作答時間） |

表 9-1‑2 view-主頁與個人資料相關

|  |  |
| --- | --- |
| 名稱 | 說明 |
| home | 渲染網站主頁，並顯示每日片語。 |
| user | 渲染使用者頁面。 |
| profile\_settings | 渲染個人設定頁面。 |
| update\_profile | 處理 POST 請求，讓使用者透過 AJAX 更新暱稱。 |
| change\_password | 處理 POST 請求，讓使用者透過 AJAX 變更密碼。 |

表 9-1‑3 view-認證與帳戶管理相關

|  |  |
| --- | --- |
| 名稱 | 說明 |
| login\_view | 處理使用者登入邏輯，驗證帳號密碼並建立會話。 |
| logout\_view | 處理使用者登出，清除會話與訊息。 |
| register\_view | 處理使用者註冊，儲存新用戶資料。 |

表 9-1‑4 view-測驗功能相關

|  |  |
| --- | --- |
| 名稱 | 說明 |
| test\_page | 渲染一個通用的測驗頁面。 |
| all\_test | 處理綜合測驗，檢查每日次數並啟動測驗。 |
| part2 | 處理 Part 2 測驗，檢查每日次數並啟動測驗。 |
| part3 | 處理 Part 3 測驗，檢查每日次數並啟動測驗。 |
| part5 | 處理 Part 5 測驗，檢查每日次數並啟動測驗。 |
| part6 | 處理 Part 6 測驗，檢查每日次數並啟動測驗。 |
| part7 | 處理 Part 7 測驗，檢查每日次數並啟動測驗。 |
| submit\_test\_answer | 使用者提交測驗答案，計算分數、發放點數並儲存結果。 |
| test\_result | 渲染測驗結果頁面。 |
| check\_test\_limit | 檢查使用者當日的測驗次數是否達上限，不啟動測驗。 |
| update\_exam\_status | 接收 AJAX 請求，更新測驗會話的狀態。 |
| exchange\_test | 處理使用者用點數兌換額外測驗次數的邏輯。 |

表 9-1‑5 view-學習與紀錄相關

|  |  |
| --- | --- |
| 名稱 | 說明 |
| record | 顯示使用者的綜合學習紀錄，包含測驗歷史、各 Part 及類別的作答表現。 |
| get\_points\_history | 提供使用者點數變動記錄的 API 接口。 |
| get\_daily\_vocabulary | 提供 API 接口，根據使用者的學習興趣推薦每日單字。 |
| mark\_word\_as\_familiar | 處理 POST 請求，將單字標記為熟悉。 |
| update\_learning\_interests | 處理 POST 請求，更新使用者的學習興趣。 |
| history\_view | 顯示使用者已看過的單字歷史清單，並提供篩選功能。 |

表 9-1‑6 view-輔助與工具相關

|  |  |
| --- | --- |
| 名稱 | 說明 |
| faq | 渲染常見問題頁面。 |
| relation\_map\_view | 渲染單字關聯圖頁面。 |
| get\_word\_relations\_by\_ai | 透過 OpenAI API 取得單字的中譯、發音和同義詞。 |
| get\_listening\_transcript | 輔助函式，取得聽力測驗文字稿。 |
| get\_taiwan\_today | 輔助函式，取得台灣時區的當前日期。 |
| get\_daily\_record\_and\_counts | 輔助函式，取得或建立使用者當日的測驗紀錄。 |
| check\_and\_update\_test\_count | 輔助函式，檢查測驗次數並在允許情況下更新。 |
| get\_learning\_suggestions | 輔助函式，根據作答表現生成學習建議。 |

表 9-1‑7 templates-主頁與主要功能頁面

|  |  |
| --- | --- |
| 名稱 | 說明 |
| home.html | 網站首頁，提供整體平台的介紹和導航。 |
| faq.html | 常見問題頁面，解答使用者可能遇到的疑問。 |
| relation\_map.html | 單字關聯地圖頁面，視覺化單字之間的關係。 |
| history.html | 歷史單字頁面，記錄使用者查詢或學習過的單字。 |
| record.html | 學習歷程頁面，展示使用者在平台上的學習數據和進度。 |
| test.html | 測驗平台主頁，提供不同測驗模式的選擇。 |
| result.html | 測驗結果頁面，顯示測驗分數、答案分析等。 |

表 9-1‑8 templates-測驗功能相關頁面

|  |  |
| --- | --- |
| 名稱 | 說明 |
| all\_test.html | 總測驗頁面，包含了所有測驗類型。 |
| part2.html | TOEIC 聽力 Part 2 測驗頁面。 |
| part3.html | TOEIC 聽力 Part 3 測驗頁面。 |
| part5.html | TOEIC 閱讀 Part 5 測驗頁面。 |
| part6.html | TOEIC 閱讀 Part 6 測驗頁面。 |
| part7.html | TOEIC 閱讀 Part 7 測驗頁面。 |

表 9-1‑9 templates-使用者管理與認證

|  |  |
| --- | --- |
| 名稱 | 說明 |
| login.html | 使用者登入頁面。 |
| register.html | 新使用者註冊頁面。 |
| profile\_settings.html | 個人設定頁面，讓使用者修改帳號資訊。 |

表 9-1‑10 templates-密碼重設流程

|  |  |
| --- | --- |
| 名稱 | 說明 |
| password\_reset\_form.html | 密碼重設表單頁面。 |
| password\_reset\_done.html | 密碼重設發送郵件通知頁面。 |
| password\_reset\_email.html | 密碼重設郵件的內容模板。 |
| password\_reset\_subject.txt | 密碼重設郵件的主旨文字檔案。 |
| password\_reset\_confirm.html | 密碼重設確認頁面，用於設定新密碼。 |
| password\_reset\_complete.html | 密碼重設完成通知頁面。 |

* 程式規格描述

|  |  |
| --- | --- |
| 程式名稱 | view-home |
| 目的 | 顯示主要資訊及每日片語 |
| 部分程式碼 | |
| def home(request):      """      渲染主頁面，並在頁面上隨機顯示每日片語，同時傳遞用戶資訊。      """      username = None      if request.user.is\_authenticated:          username = request.user.email      try:          all\_phrases = list(Phrase.objects.all())          today = date.today()          random.seed(today.day + today.month + today.year)            if len(all\_phrases) >= 3:              daily\_phrases = random.sample(all\_phrases, 3)          else:              daily\_phrases = all\_phrases            context = {              'daily\_phrases': daily\_phrases,              'user': request.user,              'username': username          }      except Exception as e:          context = {              'error\_message': f'無法載入片語：{e}',              'user': request.user,              'username': username          }          daily\_phrases = []      return render(request, "home.html", context) | |

|  |  |
| --- | --- |
| 程式名稱 | view-update\_profile |
| 目的 | 更新使用者資料 |
| 部分程式碼 | |
| @login\_required  @require\_POST  def update\_profile(request):      try:          data = json.loads(request.body)          nickname = data.get('nickname', '').strip()          if not nickname:              return JsonResponse({'success': False, 'error': '暱稱不能為空。'}, status=400)            request.user.nickname = nickname          request.user.save()          messages.success(request, '暱稱更新成功！')          return JsonResponse({'success': True})      except json.JSONDecodeError:          return JsonResponse({'success': False, 'error': '無效的 JSON 資料。'}, status=400) | |

|  |  |
| --- | --- |
| 程式名稱 | view-change\_password |
| 目的 | 使用者變更密碼 |
| 部分程式碼 | |
| @login\_required  @require\_POST  def change\_password(request):      try:          data = json.loads(request.body)          old\_password = data.get('old\_password')          new\_password1 = data.get('new\_password1')          new\_password2 = data.get('new\_password2')            # 建立 PasswordChangeForm 實例並傳入資料          form = PasswordChangeForm(user=request.user, data={              'old\_password': old\_password,              'new\_password1': new\_password1,              'new\_password2': new\_password2,          })            if form.is\_valid():              user = form.save()              update\_session\_auth\_hash(request, user)              messages.success(request, '密碼已成功變更。')              return JsonResponse({'success': True})          else:              errors = [f'{field}: {", ".join(error\_list)}' for field, error\_list in form.errors.items()]              return JsonResponse({'success': False, 'error': '; '.join(errors)}, status=400)      except json.JSONDecodeError:          return JsonResponse({'success': False, 'error': '無效的 JSON 資料。'}, status=400) | |

|  |  |
| --- | --- |
| 程式名稱 | view-login\_view |
| 目的 | 使用者登入 |
| 部分程式碼 | |
| def login\_view(request):      if request.method == 'POST':          email = request.POST.get('email')          password = request.POST.get('password')          if not email or not password:              messages.error(request, '請輸入電子郵件和密碼')              return render(request, 'login.html')          user = authenticate(request, username=email, password=password)          if user is not None:              login(request, user)              messages.success(request, '登入成功！')              return redirect('home')          else:              messages.error(request, '帳號或密碼錯誤')              return render(request, 'login.html')      return render(request, 'login.html') | |

|  |  |
| --- | --- |
| 程式名稱 | view-logout\_view |
| 目的 | 使用者登出 |
| 部分程式碼 | |
| def logout\_view(request):      """      自訂的登出視圖，確保清除所有快閃訊息。      """      logout(request)      storage = messages.get\_messages(request)      storage.used = True        messages.success(request, '您已成功登出！')      # 重新導向到登入頁面。      return redirect('login') | |

|  |  |
| --- | --- |
| 程式名稱 | view-register\_view |
| 目的 | 使用者註冊帳號 |
| 部分程式碼 | |
| def register\_view(request):      if request.method == 'POST':          form = RegisterForm(request.POST)          if form.is\_valid():              form.save()              messages.success(request, '註冊成功！請登入。')              return redirect('login')          else:              messages.error(request, '註冊失敗，請檢查表單內容。')      else:          form = RegisterForm()      return render(request, 'register.html', {'form': form}) | |

|  |  |
| --- | --- |
| 程式名稱 | view-submit\_test\_answer |
| 目的 | 接收使用者測驗答案、計算分數並回傳詳細結果與分析 |
| 部分程式碼 | |
| @csrf\_exempt  @require\_POST  def submit\_test\_answer(request):  """  接收使用者測驗答案，計算分數並回傳結果。  """  data = json.loads(request.body)  session\_id = data.get('session\_id')  answers = data.get('answers')  # 取得考試 session ---備註: 驗證 session 存在與狀態---  session = ExamSession.objects.get(session\_id=session\_id)  user = session.user  # 初始化統計  total\_questions, correct\_total = 0, 0  part\_analysis, category\_analysis, question\_details = {}, {}, []  # 逐題檢查 ---備註: 確認答案是否正確，儲存 UserAnswer---  for eq in ExamQuestion.objects.filter(exam=session.exam):  question = eq.question  selected\_option = answers.get(str(question.question\_id), '')  is\_correct = (selected\_option.lower() == question.is\_correct.lower()) if selected\_option else False  # ---備註: 累積統計、紀錄題目細節---  total\_questions += 1  if is\_correct: correct\_total += 1  # 計算分數 ---備註: 轉換為百分比---  def calc\_score(correct, total): return round((correct / total \* 100), 2) if total else 0  total\_score = calc\_score(correct\_total, total\_questions)  is\_passed = total\_score >= float(session.exam.passing\_score)  # 點數發放 ---備註: >=80分給10點, >=60分給5點---  points\_to\_award = 10 if total\_score >= 80 else (5 if total\_score >= 60 else 0)  # 儲存測驗結果 & 更新狀態 ---備註: 使用 transaction.atomic() 保護---  ExamResult.objects.create(  session=session, total\_questions=total\_questions,  correct\_answers=correct\_total, total\_score=total\_score,  is\_passed=is\_passed, completed\_at=timezone.now()  )  if points\_to\_award > 0:  user.point += points\_to\_award  user.save()  PointTransaction.objects.create(user=user, change\_amount=points\_to\_award, reason='test\_completion')  session.status = 'completed'  session.end\_time = timezone.now()  session.save()  # 回傳結果  return JsonResponse({  'success': True,  'data': {  'score': total\_score,  'correct\_answers': correct\_total,  'total\_questions': total\_questions,  'is\_passed': is\_passed,  'points\_awarded': points\_to\_award,  # ---備註: 省略詳細題目資訊, part/category 分析---  }  }) | |

|  |  |
| --- | --- |
| 程式名稱 | view-check\_test\_limit |
| 目的 | 檢查使用者當日的測驗次數是否已達上限 |
| 部分程式碼 | |
| @login\_required  @require\_POST  def check\_test\_limit(request):      """      檢查使用者當日的測驗次數是否已達上限。      """      try:          data = json.loads(request.body)          part\_number\_str = data.get('part\_number')          part\_number = int(part\_number\_str)      except (json.JSONDecodeError, ValueError, TypeError):          return JsonResponse({'status': 'error', 'message': '無效的請求數據'}, status=400)      user = request.user      today = get\_taiwan\_today()      record, \_ = DailyTestRecord.objects.get\_or\_create(user=user, date=today)      if part\_number == 0:          if record.mixed\_test\_count >= record.mixed\_test\_limit:              points\_needed = EXCHANGE\_POINTS\_FOR\_MIXED\_TEST              message = f"今日 {PART\_DICT[part\_number]} 次數已達上限 ({record.mixed\_test\_limit} 次)。"              return JsonResponse({'status': 'limit\_reached', 'message': message, 'points\_needed': points\_needed})      else:          if record.other\_part\_test\_count >= record.other\_part\_test\_limit:              points\_needed = EXCHANGE\_POINTS\_FOR\_OTHER\_PART\_TEST              test\_name = PART\_DICT.get(part\_number, f"Part {part\_number}")              message = f"今日 {test\_name} 次數已達上限 ({record.other\_part\_test\_limit} 次)。"              return JsonResponse({'status': 'limit\_reached', 'message': message, 'points\_needed': points\_needed})      return JsonResponse({'status': 'ok', 'message': '次數充足'}) | |

|  |  |
| --- | --- |
| 程式名稱 | view-all\_test |
| 目的 | 綜合測驗功能 |
| 部分程式碼 | |
| @login\_required  def all\_test(request):      user = request.user      is\_allowed, message = check\_and\_update\_test\_count(user, part\_number=0)        if not is\_allowed:          return render(request, 'home.html', {'message': message})        try:          available\_exams = Exam.objects.filter(part=0)          if not available\_exams.exists():              return render(request, 'home.html', {'message': '找不到綜合測驗。請執行管理指令以建立。'})          exam = random.choice(available\_exams)      except Exception as e:          return render(request, 'home.html', {'message': f'發生錯誤：{e}'})      # 手動處理時間戳記      utc\_now = datetime.utcnow()      taiwan\_now = utc\_now + timedelta(hours=8)      session = ExamSession.objects.create(          exam=exam,          user=user,          time\_limit\_enabled=True,          start\_time=taiwan\_now,          end\_time=taiwan\_now + timedelta(minutes=exam.duration\_minutes),          status='in\_progress',      )        exam\_questions = ExamQuestion.objects.filter(exam=exam).select\_related('question\_\_passage', 'question\_\_material').order\_by('question\_order')        if not exam\_questions.exists():          return render(request, 'error\_page.html', {'message': '綜合測驗沒有題目。'})        questions\_data = []        for eq in exam\_questions:          q = eq.question            question\_dict = {              'question\_id': str(q.question\_id),              'part': q.part,              'question\_text': q.question\_text,              'option\_a\_text': q.option\_a\_text,              'option\_b\_text': q.option\_b\_text,              'option\_c\_text': q.option\_c\_text,              'option\_d\_text': q.option\_d\_text,              'difficulty\_level': q.difficulty\_level,              'explanation': q.explanation,              'is\_correct': q.is\_correct,              'question\_category': q.get\_question\_category\_display()          }            if q.part in [2, 3] and q.material:              audio\_url = q.material.audio\_url              if audio\_url and not audio\_url.startswith(('http://', 'https://', '/')):                  audio\_url = os.path.join(settings.MEDIA\_URL, audio\_url)                question\_dict['audio\_url'] = audio\_url              question\_dict['transcript'] = q.material.transcript              question\_dict['material'] = {                  'topic': q.material.topic,                  'accent': q.material.accent,                  'listening\_level': q.material.listening\_level,              }            if q.part in [6, 7] and q.passage:              question\_dict['passage\_title'] = q.passage.title              question\_dict['passage\_content'] = q.passage.content              question\_dict['passage'] = {                  'topic': q.passage.topic,                  'word\_count': q.passage.word\_count,                  'reading\_level': q.passage.reading\_level,              }          questions\_data.append(question\_dict)        questions\_json = json.dumps(questions\_data)        context = {          'exam\_title': exam.title,          'exam\_id': str(exam.exam\_id),          'session\_id': str(session.session\_id),          'questions\_json': questions\_json,          'duration\_minutes': exam.duration\_minutes,      }        return render(request, 'all\_test.html', context) | |

|  |  |
| --- | --- |
| 程式名稱 | view-exchange\_test |
| 目的 | 供使用者利用點數兌換測驗次數 |
| 部分程式碼 | |
| @login\_required  @transaction.atomic  def exchange\_test(request):      """      處理使用者兌換額外測驗次數的 View。      """      user = request.user        if request.method != 'POST':          return JsonResponse({'status': 'error', 'message': '無效的請求方法'}, status=405)        try:          data = json.loads(request.body)          exam\_type = data.get('exam\_type')      except (json.JSONDecodeError, AttributeError):          return JsonResponse({'status': 'error', 'message': '無效的請求數據'}, status=400)        if exam\_type not in ['mixed', 'part']:          return JsonResponse({'status': 'error', 'message': '無效的測驗類型'}, status=400)        if exam\_type == 'mixed':          points\_needed = EXCHANGE\_POINTS\_FOR\_MIXED\_TEST      else:          points\_needed = EXCHANGE\_POINTS\_FOR\_OTHER\_PART\_TEST        if user.point < points\_needed:          return JsonResponse({'status': 'error', 'message': f'您的點數不足，無法兌換。需要 {points\_needed} 點。'})        user.point -= points\_needed      user.save()        PointTransaction.objects.create(          user=user,          change\_amount=-points\_needed,          reason='exam\_exchange'      )        today = get\_taiwan\_today()      record, \_ = DailyTestRecord.objects.get\_or\_create(user=user, date=today)      if exam\_type == 'mixed':          # 新增一次綜合測驗機會          record.mixed\_test\_limit += 1      else:          # 新增一次單一 Part 測驗機會          record.other\_part\_test\_limit += 1        record.save()        return JsonResponse({'status': 'success', 'message': f'成功兌換 {points\_needed} 點，已為您新增一次測驗機會！', 'current\_points': user.point}) | |

|  |  |
| --- | --- |
| 程式名稱 | view-record |
| 目的 | 使用者學習歷程頁面 |
| 部分程式碼 | |
| @login\_required  def record(request):      user = request.user      part\_names = {          2: "應答問題",          3: "簡短獨白",          5: "句子填空",          6: "段落填空",          7: "閱讀測驗",      }      # 歷史測驗紀錄      exam\_results = (          ExamResult.objects          .filter(session\_\_user=user)          .order\_by('-completed\_at')          .select\_related('session\_\_exam')      )      # 設定每頁顯示的筆數      items\_per\_page = 10      paginator = Paginator(exam\_results, items\_per\_page)      page\_number = request.GET.get('page')      page\_obj = paginator.get\_page(page\_number)      # 為每個 ExamResult 物件增加 part\_display 屬性      for result in page\_obj:          result.part\_display = result.session.exam.get\_part\_display()      # 閱讀作答情況      reading\_total = UserAnswer.objects.filter(session\_\_user=user, question\_\_question\_type='reading').count()      reading\_correct = UserAnswer.objects.filter(session\_\_user=user, question\_\_question\_type='reading', is\_correct=True).count()      # 聽力作答情況      listening\_total = UserAnswer.objects.filter(session\_\_user=user, question\_\_question\_type='listen').count()      listening\_correct = UserAnswer.objects.filter(session\_\_user=user, question\_\_question\_type='listen', is\_correct=True).count()      # 計算百分比      reading\_progress = int((reading\_correct / reading\_total) \* 100) if reading\_total else 0      listening\_progress = int((listening\_correct / listening\_total) \* 100) if listening\_total else 0      # 總學習時數      total\_answers = UserAnswer.objects.filter(session\_\_user=user).count()      study\_hours = round(total\_answers / 60, 1)      part\_performance = {}      part\_numbers = [2, 3, 5, 6, 7] # 你們的測驗Part        for part in part\_numbers:          part\_answers = UserAnswer.objects.filter(              session\_\_user=user,              question\_\_part=part          )          total\_in\_part = part\_answers.count()          correct\_in\_part = part\_answers.filter(is\_correct=True).count()          percentage\_in\_part = int((correct\_in\_part / total\_in\_part) \* 100) if total\_in\_part else 0            part\_display\_name = part\_names.get(part, f'Part {part}')            if total\_in\_part > 0: # 只顯示有作答紀錄的 Part              part\_performance[part] = {                  'display\_name': part\_display\_name,                  'total': total\_in\_part,                  'correct': correct\_in\_part,                  'percentage': percentage\_in\_part,              }      category\_performance = {}      category\_choices\_dict = dict(QUESTION\_CATEGORY\_CHOICES)      for category\_key, category\_display\_name in QUESTION\_CATEGORY\_CHOICES:          category\_answers = UserAnswer.objects.filter(              session\_\_user=user,              question\_\_question\_category=category\_key          )          total\_in\_category = category\_answers.count()          correct\_in\_category = category\_answers.filter(is\_correct=True).count()          percentage\_in\_category = int((correct\_in\_category / total\_in\_category) \* 100) if total\_in\_category else 0            # 只有在有作答紀錄時才加入字典          if total\_in\_category > 0:              category\_performance[category\_key] = {                  'display\_name': category\_display\_name,                  'total': total\_in\_category,                  'correct': correct\_in\_category,                  'percentage': percentage\_in\_category,              }      context = {          'user': user,          'page\_obj': page\_obj,          'reading\_progress': reading\_progress,          'listening\_progress': listening\_progress,          'study\_hours': study\_hours,          'reading\_total': reading\_total,          'reading\_correct': reading\_correct,          'listening\_total': listening\_total,          'listening\_correct': listening\_correct,          'category\_performance': category\_performance,          'part\_performance': part\_performance,          'learning\_suggestions': get\_learning\_suggestions(category\_performance),      }      return render(request, 'record.html', context) | |

|  |  |
| --- | --- |
| 程式名稱 | view-get\_daily\_vocabulary |
| 目的 | 使用者每日領取三個單字卡 |
| 部分程式碼 | |
| @login\_required  @require\_http\_methods(["GET"])  def get\_daily\_vocabulary(request):      """      根據使用者的學習興趣推薦單字。      推薦邏輯：      1. 優先推薦 1 個「已經看過但未標記為熟悉」的舊單字。      2. 接著推薦 2 個「從未看過」的新單字。      使用者每天最多只能領取一次。      """      user = request.user      today\_date = timezone.now().date()        already\_sent\_today = UserVocabularyRecord.objects.filter(          user=user,          sent\_time\_\_date=today\_date      ).exists()      if already\_sent\_today:          # 如果今天已經有任何單字被發送過，則不發送新單字          return JsonResponse({'message': '你今天已經領取過單字囉！可至歷史單字區查看單字紀錄', 'words': []})        # 獲取使用者已熟悉的單字 ID      familiar\_words\_ids = UserVocabularyRecord.objects.filter(          user=user,          is\_familiar=True      ).values\_list('word\_id', flat=True)      # 獲取所有已看過但未熟悉的單字 ID (弱點單字)      weak\_words\_ids = UserVocabularyRecord.objects.filter(          user=user,          is\_familiar=False      ).values\_list('word\_id', flat=True)        # 獲取使用者設定的學習興趣      user\_interests = user.learning\_interests.split(',') if hasattr(user, 'learning\_interests') and user.learning\_interests else []        # 根據興趣篩選單字      vocabulary\_queryset = DailyVocabulary.objects.all()      if user\_interests:          vocabulary\_queryset = vocabulary\_queryset.filter(related\_category\_\_in=user\_interests)      # 獲取所有尚未熟悉的單字 ID (包含弱點和從未看過的)，並應用興趣篩選      all\_unfamiliar\_words\_ids = vocabulary\_queryset.exclude(          id\_\_in=familiar\_words\_ids      ).values\_list('id', flat=True)      # 獲取所有從未看過的單字 ID (排除所有已看過的單字)，並應用興趣篩選      unseen\_words\_ids = vocabulary\_queryset.exclude(          id\_\_in=familiar\_words\_ids      ).exclude(          id\_\_in=weak\_words\_ids      ).values\_list('id', flat=True)      words\_to\_get\_ids = []      # 1. 優先推薦 1 個舊單字 (弱點單字)，並確保它在興趣範圍內      weak\_words\_in\_interest = [wid for wid in weak\_words\_ids if wid in all\_unfamiliar\_words\_ids]      if weak\_words\_in\_interest:          weak\_word\_id = random.choice(weak\_words\_in\_interest)          words\_to\_get\_ids.append(weak\_word\_id)      # 2. 接著推薦 2 個新單字，並確保它在興趣範圍內      num\_new\_words = 3 - len(words\_to\_get\_ids)      if unseen\_words\_ids and num\_new\_words > 0:          new\_word\_candidates = [wid for wid in unseen\_words\_ids if wid not in words\_to\_get\_ids]          num\_to\_sample = min(len(new\_word\_candidates), num\_new\_words)          words\_to\_get\_ids.extend(random.sample(new\_word\_candidates, num\_to\_sample))      # 如果總數仍不足，則從所有未熟悉的單字中隨機補足      if len(words\_to\_get\_ids) < 3:          remaining\_slots = 3 - len(words\_to\_get\_ids)          remaining\_candidates = [wid for wid in all\_unfamiliar\_words\_ids if wid not in words\_to\_get\_ids]          num\_to\_sample = min(len(remaining\_candidates), remaining\_slots)          if remaining\_candidates:              words\_to\_get\_ids.extend(random.sample(remaining\_candidates, num\_to\_sample))        if not words\_to\_get\_ids:          # 如果使用者已經學完所有該興趣下的單字，或者沒有單字可推薦          message = '恭喜你！所有單字都已完成學習，或目前沒有符合你興趣的單字了。'          return JsonResponse({'message': message, 'words': []})        new\_words = DailyVocabulary.objects.filter(id\_\_in=words\_to\_get\_ids)      words\_data = []      for word in new\_words:            UserVocabularyRecord.objects.update\_or\_create(              user=user,              word=word,              defaults={                  'last\_viewed': timezone.now(),              }          )          words\_data.append({              'word\_id': word.id,              'word': word.word,              'pronunciation': word.pronunciation,              'part\_of\_speech': word.part\_of\_speech,              'translation': word.translation,              'example\_sentence': word.example\_sentence,              'example\_translation': word.example\_translation,              'difficulty\_level': word.difficulty\_level,              'related\_category': word.related\_category,          })      return JsonResponse({'message': '這是你今天的單字清單：', 'words': words\_data}) | |

|  |  |
| --- | --- |
| 程式名稱 | view-get\_word\_relations\_by\_ai |
| 目的 | 生成單字關聯地圖 |
| 部分程式碼 | |
| def get\_word\_relations\_by\_ai(request, word):      try:            prompt\_text = f"""          Please provide information for the English word: {word}          Return JSON only, strictly following this format:          {{              "translation": "中文翻譯，請使用繁體中文",              "pronunciation": "KK or IPA phonetic transcription",              "synonyms": [                  {{                      "word": "synonym1",                      "synonyms": [                          {{                              "word": "subsyn1",                              "synonyms": ["subsub1", "subsub2"]                          }},                          ...                      ]                  }},                  ...              ]          }}          Rules:          - "translation" must be Traditional Chinese only.          - "pronunciation" should be phonetic transcription (KK or IPA).          - "synonyms" must be English words only.          - Each level can have 0~3 sub-synonyms.          - Max depth = 3 levels (center -> top -> second -> third).          - Do not include Chinese in synonyms.          """            response = client.chat.completions.create(              model="gpt-4o-mini",              messages=[{"role": "user", "content": prompt\_text}],              response\_format={"type": "json\_object"}          )          ai\_text = response.choices[0].message.content.strip()          ai\_data = json.loads(ai\_text)          response\_data = {              "source\_word": word,              "translation": ai\_data.get("translation", ""),              "pronunciation": ai\_data.get("pronunciation", ""),              "synonyms": ai\_data.get("synonyms", []) if isinstance(ai\_data.get("synonyms"), list) else []          }          return JsonResponse(response\_data)      except openai.APIError as e:          return JsonResponse({              "source\_word": word,              "translation": "",              "pronunciation": "",              "synonyms": [],              "error": f"OpenAI API 錯誤: {e}"          }, status=500)      except (json.JSONDecodeError, IndexError, KeyError) as e:          return JsonResponse({              "source\_word": word,              "translation": "",              "pronunciation": "",              "synonyms": [],              "error": f"無法解析 AI 回應: {e}"          }, status=500) | |

## 其他附屬之各種元件

表 9-2‑1 套件-核心框架與伺服器

|  |  |  |
| --- | --- | --- |
| 套件名稱 | 版本 | 說明 |
| Django | 5.2.1 | 主要的 Web 框架，提供路由、模板和 ORM 功能。 |
| gunicorn | 23.0.0 | 正式環境用的 Web 伺服器，負責運行 Django 應用程式。 |
| asgiref | 3.8.1 | ASGI 介面，讓 Django 支援非同步 (async/await) 模式。 |
| gevent | 25.5.1 | 協同多工函式庫，用於提高伺服器的並發處理能力。 |
| greenlet | 3.2.2 | gevent 的底層依賴，用於輕量級線程。 |
| cors | 1.0.1 | 處理跨來源資源共享 (CORS) 政策。 |

表 9-2‑2 套件-資料庫與資料處理

|  |  |  |
| --- | --- | --- |
| 套件名稱 | 版本 | 說明 |
| mysqlclient | 2.2.7 | Python 與 MySQL 資料庫之間的連接驅動程式。 |
| mysql-connector-python | 9.3.0 | 另一個 Python 的 MySQL 連接器，通常用於純 Python 環境。 |
| sqlparse | 0.5.3 | 解析和格式化 SQL 語句，方便日誌記錄或偵錯。 |
| pydantic | 2.11.9 | 強大的資料驗證和解析函式庫。 |
| pydantic\_core | 2.33.2 | pydantic 的核心，提供底層的資料驗證能力。 |
| jiter | 0.11.0 | 針對 JSON 資料進行高效能解析。 |
| tzdata | 2025.2 | 提供全球時區資料，確保時間處理的準確性。 |
| zope.event | 5 | 提供事件通知功能，幫助應用程式解耦。 |
| zope.interface | 7.2 | 定義物件介面，用於大型應用程式的架構設計。 |

表 9-2‑3 套件- API 溝通與網路請求

|  |  |  |
| --- | --- | --- |
| 套件名稱 | 版本 | 說明 |
| openai | 1.107.2 | OpenAI API 的官方函式庫，用於整合 AI 功能。 |
| httpx | 0.28.1 | 現代化的 HTTP 客戶端，支援非同步請求。 |
| requests | 2.32.3 | 簡單易用的 HTTP 函式庫，用於發送網頁請求。 |
| requests-file | 2.1.0 | 讓 requests 能處理本地檔案路徑。 |
| httpcore | 1.0.9 | httpx 的底層依賴，處理低階 HTTP 協議。 |
| h11 | 0.16.0 | 處理 HTTP/1.1 協議，是 httpcore 的依賴。 |
| python-dotenv | 1.1.1 | 從 .env 檔案載入環境變數。 |
| certifi | 2025.8.3 | 處理 SSL 憑證，確保網路連線安全。 |
| idna | 3.1 | 處理國際化域名。 |
| urllib3 | 2.4.0 | 處理 URL 請求，是 requests 的依賴。 |
| PySocks | 1.7.1 | 處理 SOCKS 代理協議。 |
| tldextract | 5.3.0 | 從網址中提取頂級域名。 |

表 9-2‑4 套件- 輔助與開發工具

|  |  |  |
| --- | --- | --- |
| 套件名稱 | 版本 | 說明 |
| tqdm | 4.67.1 | 顯示進度條，用於長時間運行的任務。 |
| colorama | 0.4.6 | 讓終端機輸出有顏色。 |
| future | 1.0.0 | 提供 Python 2 和 3 的相容性。 |
| distro | 1.9.0 | 取得作業系統版本資訊。 |
| filelock | 3.18.0 | 提供進程間的檔案鎖定。 |
| Jinja2 | 3.1.6 | 模板引擎，用於生成 HTML 頁面。 |
| MarkupSafe | 3.0.2 | 確保 Jinja2 渲染時的字串安全。 |
| setuptools | 80.8.0 | Python 專案打包工具。 |
| wheel | 0.45.1 | Python 的打包格式。 |
| packaging | 25 | 管理套件版本和相依性。 |
| sniffio | 1.3.1 | 偵測非同步後端。 |
| charset-normalizer | 3.4.2 | 偵測文字編碼。 |
| sqlparse | 0.5.3 | 解析與格式化 SQL 語法。 |

表 9-2‑5 套件- 型別提示與程式碼品質

|  |  |  |
| --- | --- | --- |
| 套件名稱 | 版本 | 說明 |
| typing\_extensions | 4.15.0 | 提供 Python 新版本中的型別提示功能。 |
| annotated-types | 0.7.0 | 支援更進階的型別提示。 |
| typing-inspection | 0.4.1 | 檢查型別提示的工具。 |

# 測試模型

# 操作手冊

# 使用手冊

# 感想

# 參考資料

* [TOEIC® Listening and Reading Test](https://www.toeic.com.tw/toeic/listening-reading/about/content/)
* [【多益單字總整理】ー新制多益必考單字大全!](https://tw.amazingtalker.com/blog/zh-tw/zh-eng/10141/)
* [全新制多益TOEIC必考單字3,000](https://www.17buy.com.tw/books/download.php?op=1&cid=20)