包裹追蹤與計費系統

組員: 軟體二 方思涵 411377023、軟體二 曾湧庭 411377022、
軟體二 許云馨 411377016、數學四 蔡宗翰 411131225

目錄

1. 專案規劃 (Project Planning)

1.1 專案背景

隨著電商與全球物流量快速成長，黑貓、FedEx、UPS、DHL 等物流公司每日需處理大量包裹，對資訊系統的效率、即時性與準確性提出更高要求；客戶也期待透明且可即時查詢的配送資訊。為因應這些需求，本專案開發一套整合性的包裹追蹤與計費系統，模擬物流業者的核心後端流程，提供包裹狀態追蹤、配送事件紀錄、計費與帳務、客戶查詢及角色權限控管等功能，以提升物流作業效率並降低人工錯誤。

1.2 專案目標

一、客戶與包裹資料管理

- 提供完整的客戶資料管理（基本資訊、帳單偏好、客戶類型）。
- 支援包裹寄件登錄、屬性紀錄（重量、尺寸、內容物價值）。
- 系統自動產生唯一追蹤碼，確保包裹識別一致性。

二、物流事件追蹤與狀態查詢

- 記錄包裹生命週期中的所有物流事件（收件、分揀、運輸、外送、送達）。
- 提供即時狀態查詢與完整歷史事件紀錄。
- 提升物流資訊透明度與可追溯性。

三、計費與帳務管理

- 依重量、距離、服務類型等自動計算運費。
- 支援不同付款方式並整合基本帳務流程。
- 降低人工計費錯誤並提升作業效率。

四、角色權限與系統安全

- 提供角色權限控管（客戶、作業人員(客服、倉儲人員、管理者)、駕駛員）。
- 限制資料讀寫範圍，確保系統與資訊安全。

五、系統操作介面與管理分析

- 提供 Web/Console 操作介面，以提升使用與管理便利性。
- 提供報表與統計功能（出貨量、延誤率、退貨率等）。
- 協助管理者進行營運分析與決策支持。

1.3 專案組織與分工

本專案採用**敏捷開發模式**，組員分工如下：

角色	姓名	主要職責 (Key Responsibilities)
PM & 系統分析	**許云馨**	專案時程控管、UML 模型設計 (類別圖/序列圖)、權限控管模組規劃與實作、期末報告整合
系統基建 & 架構	**曾湧庭**	系統基礎建設 (Github)、客戶管理 API、系統架構圖繪製、文件撰寫
QA & 核心業務& 後端	**方思涵**	核心商業邏輯實作 (包裹/追蹤)、核心商業邏輯、資料庫(models.py、db_operations.py)、Pytest 測試。
全端整合 & 金流	**蔡宗翰**	計費模組、金流模擬、前端介面 (UI/UX) 開發與優化、Demo 製作。

1.4 時程規劃

週次	階段	內容	里程碑
第一週	專案規劃、需求分析	明確專案目標、利害關係人；撰寫使用案例與需求說明	**M1**：需求確認
第二週	系統設計（UML、架構）	繪製完整類別圖、至少一項序列圖與系統架構圖	**M2**：系統設計完成
第三~四週	系統開發	前後端開發、資料庫建置；建立 Git 儲存庫與分支策略，進行版本控管	**M3**：核心功能完成
第五週	品質管理與測試	撰寫單元測試、執行整合測試與缺陷修正	**M4**：通過主要測試
第六週	測試案例驗證、展示說明	彙整測試案例與結果，撰寫專案文件，準備簡報與展示影片	**M5**：專題完成

1.5 成本估算

1. 敏捷式 Story Points（相對估算）

功能模組	複雜度 (Story Points)	估算等級	轉換工時 (預估)	說明
客戶管理模組	**5 SP**	中	40 hr	包含 CRUD 基本操作，邏輯單純。
包裹管理模組	**8 SP**	中高	64 hr	需處理追蹤碼生成演算法與資料庫關聯。
追蹤事件模組	**13 SP**	高	104 hr	**(核心難點)** 涉及時間軸排序、狀態變更與大量資料寫入。
計費與金流模組	**8 SP**	中高	64 hr	需串接外部模擬金流，並處理費率計算公式。
帳單與報表模組	**5 SP**	中	40 hr	彙整歷史數據產出報表，偏向查詢功能。
權限與安全模組	**3 SP**	低	24 hr	使用 JWT 標準驗證，現有框架支援度高。
前端介面整合	**8 SP**	中高	64 hr	需整合上述所有 API，並處理 RWD 切版。
總計	**50 SP**	—	**400 hr**	**做為人力成本計算基礎**

2. 人力成本估算表

本專案成本估算分為兩部分：一是基於學生團隊實際投入時間的「內部開發成本（學術版）」，**二是模擬若將此需求外包給專業軟體公司開發的**「市場行情估算（商業版）」。

A. 內部開發成本（學術版）

此估算反映團隊成員在 6 週內實際投入的時間成本（Opportunity Cost）。假設每位成員每週投入約 20 小時，採實習生/初階助理時薪計算。

1. 人力成本估算表 (修正版)

依據 6 週全端分工表進行工時重算：

角色職責	對應組員	預估投入工時 (6 週)	時薪 (NT$)	小計 (NT$)	工作內容說明 & 時數分析
PM & 系統分析	許云馨	**100 hr**	$250	$25,000	專案時程控管、UML 設計、權限模組實作、期末報告彙整。
系統基建 & 架構	曾湧庭	**80 hr**	$250	$20,000	系統基礎建設、客戶 API、架構文件。
QA & 核心業務& 後端	方思涵	**120 hr**	$250	$30,000	核心商業邏輯、資料庫(models.py、db_operations.py)、Pytest 測試。
全端整合 & 金流	蔡宗翰	**120 hr**	$250	$30,000	計費模組、前端 UI 開發、系統整合，Demo 製作。
總計	**4** 人	**420 hr**	—	**$105,000**	**總人力開發成本**

2. 非人力成本 (基礎建設)

項目	成本 (NT$)	說明
開發工具	$0	使用 VS Code
版本控制	$0	GitHub Free Tier
雲端部署 (Demo)	$0	不會實際架設
溝通協作	$0	Discord, Line, Google Meet
總計	**$0**	**X**

B. 業界外包報價模擬（商業版）

若客戶（物流公司）將此需求發包給台灣中小型軟體接案公司，成本結構將包含：資深工程師薪資、公司營運管理費（約 20~30%）、稅務與利潤。

1. 業界人力單價參考 (2024 年行情)

- **資深系統分析師 (SA)**: $1,200 ~ $1,800 / hr

- 後端工程師 **(Backend)**: $1,000 ~ $1,500 / hr
- 前端工程師 **(Frontend)**: $1,000 ~ $1,500 / hr
- 專案經理 **(PM)**: $800 ~ $1,200 / hr

2. 外包專案報價單 **(模擬)**

項目	預估工時	平均單價 (NT$)	總價 (NT$)	備註
1. 專案管理 **(PM)**	40 hr	$1,000	$40,000	包含需求訪談、進度匯報、驗收會議
2. 系統分析與設計 **(SA)**	60 hr	$1,500	$90,000	資料庫設計、API 規格書、架構規劃
3. 後端 **API** 開發	120 hr	$1,200	$144,000	包含金流串接、物流邏輯、權限控管
4. 前端 **Web/App** 開發	120 hr	$1,200	$144,000	RWD 響應式網頁、駕駛員介面
5. 系統測試 **(QA)**	40 hr	$800	$32,000	整合測試、壓力測試、部署
6. 伺服器與網域 **(1** 年**)**	—	—	$12,000	AWS/GCP 基礎費用估算
小計	**380 hr**	—	**$462,000**	—
營業稅 **(5%)**	—	—	**$23,100**	—
總報價	—	—	**$485,100**	約新台幣 **48.5** 萬元

C. 成本效益比較與分析 **(Comparative Analysis)**

我們將 **學術開發** 與 **業界外包** 進行對比，分析其中的差異：

比較項目	學生專案 **(NT$ 10.5** 萬**)**	業界專案 **(NT$ 48.5** 萬**)**	差異原因分析

人力單價	$250 / hr	$1,000+ / hr	業界需負擔勞健保、辦公室租金、軟體授權費及公司利潤。
技術深度	功能實作為主	效能優化、資安	業界專案需考慮高併發 (High Concurrency) 與嚴格資安標準，開發難度較高。
維運承諾	無 (期末結束)	保固 1 年	業界報價通常包含上線後的 Debug 與維護服務。
文件完整度	基礎文件	完整規格書	業界需交付完整的 API 文件 (Swagger) 與操作手冊。

結論

本專案透過學生團隊自行開發，雖然在技術成熟度上不及業界資深團隊，但以 **NT$ 105,600** 的隱性成本，完成了市值約 **NT$ 485,100** 的系統雛形。這顯示了本專案具有極高的**投資報酬率 (ROI)**，且對於物流中小企業而言，是一套具備高性價比的 MVP (最小可行性產品)解決方案。

---

2. 專案需求說明 (使用案例)

本章節透過使用案例圖(Use Case Diagram)與詳細描述表，定義本系統之功能範疇與使用者互動流程。

2.1 使用案例詳細說明【附錄 A：使用案例圖】

依據圖示，各模組之詳細功能定義如下表所示：

功能模組	使用案例名稱	主要執行角色	功能簡要說明
客戶管理	建立/註冊帳號	客戶、員工	允許新用戶註冊帳號，或由員工協助建立客戶檔案。
	設定客戶資料	客戶、員工	維護基本資料（電話、地址）及帳單偏好設定。
	查看自身貨件	客戶	(隱私保護) 客戶登入後，僅能查詢自己寄出或收到的包裹。

功能模組	使用案例名稱	主要執行角色	功能簡要說明
包裹管理	建立包裹	客戶、員工、倉儲	填寫寄件資訊後，系統自動產生唯一追蹤編號 **(TRK-xxx)**。
	管理服務類型	管理員	設定標準速遞、隔夜達等不同服務的費率參數。
	匯出 **Excel** 報表	員工、管理員	**(**後台功能**)** 將包裹清單或物流紀錄匯出為 **Excel** 檔案以供備份。
物流追蹤	更新物流狀態	駕駛員、倉儲	掃描單號更新狀態（如：已攬收、運輸中），並記錄車輛 **ID** 與倉儲 **ID**。
	查詢狀態與歷史	所有角色	依追蹤碼查詢包裹的完整運送時間軸 **(Timeline)**。
	進階搜尋	員工、管理員	支援依 「車輛編號」 或 「倉儲編號」 篩選特定批次的包裹。
計費與付款	試算運費	客戶、系統	依重量與服務類型，即時計算應付運費。
	付款確認	客戶	確認金額無誤後進行付款，系統記錄「付款完成」事件。
權限安全	角色存取控制	管理員、系統	透過 **JWT Token** 驗證身分，確保客戶無法執行員工專屬功能（如修改狀態）。

---

3. 系統模型與架構 (System Architecture)

3.1 系統架構設計

本系統採用 **Python Flask 輕量級網頁框架**，搭配 **models.py**、**db_operations.py** 作為資料儲存後端，實現快速開發與部署的 MVP (最小可行性產品) 架構，詳細架構圖請參閱**【附錄 B：系統架構圖】**。

- **前端層 (Frontend)**：使用 HTML5, CSS3, JavaScript，透過 Flask Jinja2 樣板引擎渲染畫面。
- **應用層 (Backend)**：使用 Python Flask 處理 HTTP 請求、路由控制、JWT 身分驗證與商業邏輯運算。

- **資料層 (Data Layer)**:透過 **models.py**、**db_operations.py** 檔案進行讀寫操作，方便資料檢視與備份。

3.2 系統模型 (UML)

3.2.1 類別圖 (Class Diagram)

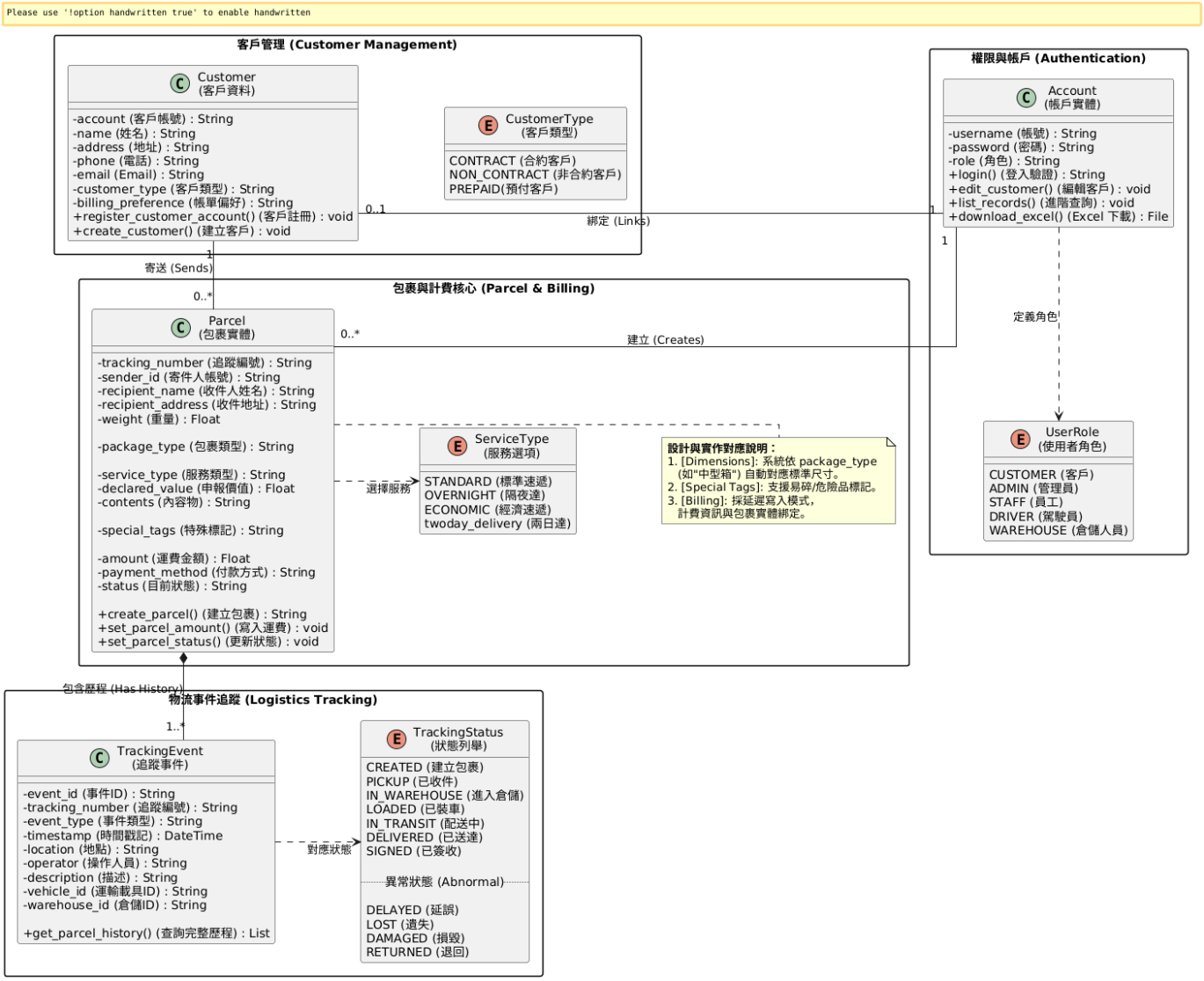


圖 3-1 系統類別圖：展示了 Account, Customer, Parcel, TrackingEvent 之間的關聯，以及透過資料庫進行存取的設計。

3.2.2 系統序列圖說明 (System Sequence Diagrams)

本系統針對核心業務流程設計了詳細的時序互動模型，以確保前後端與資料庫之間的邏輯正確性。完整序列圖請參閱報告後方之**【附錄 C】**，以下說明各關鍵流程之設計邏輯：

1. **新增客戶註冊流程 (User Registration)**
    - **參閱：附錄圖 C-1**

- **流程說明**：當新用戶透過前端介面提交註冊資訊（帳號、密碼、姓名、電話等）時，後端會先查詢資料庫檢查該帳號是否已存在。若帳號唯一，系統將會建立新的客戶資料物件，並將其寫入 Excel 的 Accounts 與 Customers 資料表，完成註冊程序。

2. **建立包裹流程 (Create Parcel)**
   - **參閱：附錄圖 C-2**
   - **流程說明**：此為自動化核心流程。當使用者提交寄件資訊後，後端 app.py 會自動依據當前日期與隨機碼生成唯一的 **追蹤編號 (Tracking Number)**，並依序執行兩個寫入動作：(1) 儲存包裹主檔、(2) 自動寫入第一筆「建立包裹」的物流事件，確保資料與狀態的一致性。
3. **更新物流狀態流程 (Update Status)**
   - **參閱：附錄圖 C-3**
   - **流程說明**：此功能設有嚴格的角色檢查，僅限駕駛員與員工使用。當更新狀態（如：已攬收、運輸中）時，Payload 需同時包含**車輛編號 (Vehicle ID)**或**倉儲編號 (Warehouse ID)**。系統會將這些輔助資訊與狀態變更一同寫入 TrackingEvent 資料表，以支援後續的進階物流追蹤。
4. **查詢包裹狀態與歷程 (Track Parcel History)**
   - **參閱：附錄圖 C-4**
   - **流程說明**：系統接收查詢請求後，會從 TrackingEvent 資料表中撈取該追蹤編號的所有歷史節點，並依照時間戳記 (Timestamp) 進行排序。回傳的 JSON 結構包含完整的時間軸數據，供前端繪製物流進度條。
5. **查詢用戶所有包裹紀錄 (List User Records)**
   - **參閱：附錄圖 C-5**
   - **流程說明**：此流程包含關鍵的**隱私過濾邏輯 (Privacy Filter)**。
     - 若操作者為**客戶**：系統會強制篩選 sender_id，確保客戶僅能看見自己寄出的包裹。
     - 若操作者為**員工/管理員**：則可查看系統內所有包裹，或進行進階條件篩選。
6. **付款處理流程 (Process Payment)**
   - **參閱：附錄圖 C-6**
   - **流程說明**：前端計算運費並經使用者確認後，呼叫付款 API。後端接收請求後，除了更新包裹主檔的付款金額與狀態外，亦會觸發稽核機制，寫入一筆「付款完成」的事

件紀錄，確保金流操作具有不可否認性。

7. **權限控制流程 (RBAC Flow)**
    - **參閱：附錄圖 C-7**
    - **流程說明**：展示系統如何透過 Python Decorator (@token_required) 攔截所有 API 請求。系統首先驗證 JWT Token 的有效性，接著檢查 Token 內的 role 欄位。若一般客戶嘗試執行員工專屬功能（如更新物流狀態），系統將直接回傳 403 Forbidden 錯誤，阻擋越權存取。

---

4. 組態管理 (Configuration Management)

本專案使用 **Git** 進行版本控制，並託管於 **GitHub**。

- **分支策略**：採用 Feature Branch Workflow。主分支(main)保持穩定，開發新功能時建立 feature/xxx 分支，測試通過後合併。
- **忽略檔案 (.gitignore)**：排除 __pycache__, .env, 以及含有真實個資的 logistics_db.xlsx，確保資安與儲存庫整潔。

---

5. 品質管理 (Quality Management)

為確保系統具備良好的可讀性、可擴充性與維護性，本專案在開發過程中嚴格遵循 Python 社群標準（PEP 8）與軟體工程最佳實踐。重點聚焦於程式碼品質、命名一致性、模組化設計與錯誤防護機制。

5.1 程式碼命名規範 (Naming Conventions)

本專案嚴格統一命名風格，降低多人協作時的認知落差與溝通成本。

1. **變數命名原則 (Variables)**
    - **風格標準：**全採用 Python 官方建議的 小寫蛇形命名法 (snake_case)，提升程式可讀性。
    - **實例說明：**使用 tracking_number、sender_id (O)；嚴格禁止 Java 風格的 trackingNumber 或 SenderID (X)。
    - **語意明確性：**拒絕過度縮寫，變數名稱需具備自解釋能力 (Self-documenting)。例如使用 recipient_address 而非 addr。

- **布林值命名：**布林變數一律以 is_ 或 has_ 等語意動詞開頭（如 is_admin, has_permission），以利邏輯判斷。

2. **函式與方法命名 (Functions & Methods)**
   - **結構定義：**採用「動詞 + 名詞」的結構，清楚表達函式行為。
   - **動詞一致性：**針對資料庫操作定義統一動詞，確保語意直觀：
     - 新增資料：統一使用 append_* (如 append_parcel)。
     - 查詢資料：統一使用 read_* 或 find_* (如 read_customers)。
     - 更新資料：統一使用 update_* (如 update_parcel_amount)。
   - **API 對應：**後端函式名稱與 API 路由 (Route) 語意保持一致，例如 /api/parcels 對應 create_parcel()，降低維護理解成本。
3. **常數與設定值 (Constants)**
   - **風格標準：**全域常數一律使用全大寫 + 底線 (UPPER_CASE)。
   - **實例說明：**例如 EXCEL_FILE、SECRET_KEY。
   - **集中管理：**將關鍵設定值集中定義於程式開頭，避免散落於程式邏輯中，降低修改風險。

5.2 程式碼結構與模組化設計 (Modularity)

本系統採用關注點分離 (Separation of Concerns) 原則，將不同職責的程式碼拆分至不同模組，避免「神一般的類別 (God Class)」出現。

- **控制器層 (Controller - app.py)：**專注於處理 HTTP 請求、路由分發、JWT 權限驗證與回應格式化，不包含底層資料操作細節。
- **資料存取層 (DAO - excel_db.py)：**專注於 Excel 檔案的讀寫操作與資料格式轉換。
- **單一職責原則 (SRP)：**每個模組與函式僅負責單一功能，例如資料處理與格式轉換邏輯分離，提升單元測試的可行性。

5.3 程式碼可讀性與維護性 (Readability & Maintainability)

1. **關鍵邏輯註解：**在複雜邏輯區塊（如計費運算、權限判斷）加上註解，重點說明「為什麼這樣做 (Why)」而非僅描述語法。
2. **函式長度控制：**遵循「一個函式只做一件事」的原則，控制函式長度，避免過度巢狀的 if-else 結構。

3. **避免魔術數字 (Magic Numbers)**：將費率、時間限制等數值定義為具名變數或常數，例如避免直接在程式中寫 * 60，而是定義 UNIT_PRICE = 60，提高程式碼的可讀性與修改彈性。

5.4 錯誤處理與防呆設計 (Error Handling)

為確保系統運作的強健性 (Robustness)，API 實作了以下防護機制：

- **輸入驗證**：所有 API 均檢查必要欄位是否存在 (Required Fields Check)，並對數值欄位（如金額、重量）進行型別檢查與轉換，防止系統崩潰。
- **統一錯誤回應**：當發生例外狀況時，系統攔截錯誤並回傳一致的 JSON 格式（如 { "error": "錯誤訊息" }）及正確的 HTTP 狀態碼 (400, 401, 403, 500)，避免前端介面因後端錯誤而白屏。
- **服務不中斷**：透過 try-except 機制包覆關鍵邏輯，確保單一請求的失敗不會導致整個伺服器停機。

5.5 前後端命名一致性 (Consistency)

為減少前後端整合時的轉換成本，本專案在 HTML 前端 與 Python 後端 API 之間建立了嚴格的欄位對照標準。

- **統一欄位名稱**：前端 JSON Payload 的 Key 與後端變數名稱保持一致（例如均使用 tracking_number, sender_id）。
- **資料結構對齊**：確保前端預期的回傳格式與後端輸出結構吻合，大幅降低整合測試時的除錯時間。

5.6 版本控管流程 (Version Control)

本專案使用 Git 進行版本控制，並遵循以下提交原則以維護程式庫品質：

- **原子性提交 (Atomic Commits)**：每次 commit 僅包含單一功能的修改，避免一次提交過多不相關的檔案。
- **語意化訊息(Commit Message):** 清楚描述修改的「目的」與「內容」，以利日後追溯。

6.系統測試與驗證 (System Testing)

本系統開發語言為 Python，因此測試階段採用 Python 標準自動化測試框架 **pytest** 進行單元測試 (Unit Testing) 與整合測試 (Integration Testing)，並搭配 **pytest-cov** 進行程式碼覆蓋率分析，以確保核心商業邏輯正確且系統運作穩定。

6.1 測試環境與工具

- **測試框架**：pytest (取代 Java 的 JUnit)
- **覆蓋率分析**：pytest-cov
- **模擬工具**：unittest.mock (用於模擬資料庫錯誤或外部依賴)
- **HTTP 客戶端**：Flask Test Client (用於模擬 API 請求)
- **測試資料庫**：測試執行時自動切換至獨立的 test_logistics_db.xlsx，測試結束後自動重置，避免汙染正式營運資料。

6.2 測試策略

本專案採用四層測試策略，確保系統品質：

1. **單元測試**：針對運費計算、密碼雜湊、Token 生成等單一功能進行驗證。
2. **整合測試**：模擬完整業務流程（如：建立包裹 → 倉儲收件 → 配送 → 送達）。
3. **安全性測試**：驗證 JWT Token 簽章、過期機制及 RBAC (角色權限控管)。
4. **邊界值測試**：針對負數金額、空值輸入、極大數值進行防呆測試。

6.3 測試案例規劃 (Test Cases)

本專案針對關鍵功能模組設計了詳細的測試案例，以下列出最具代表性之核心案例：

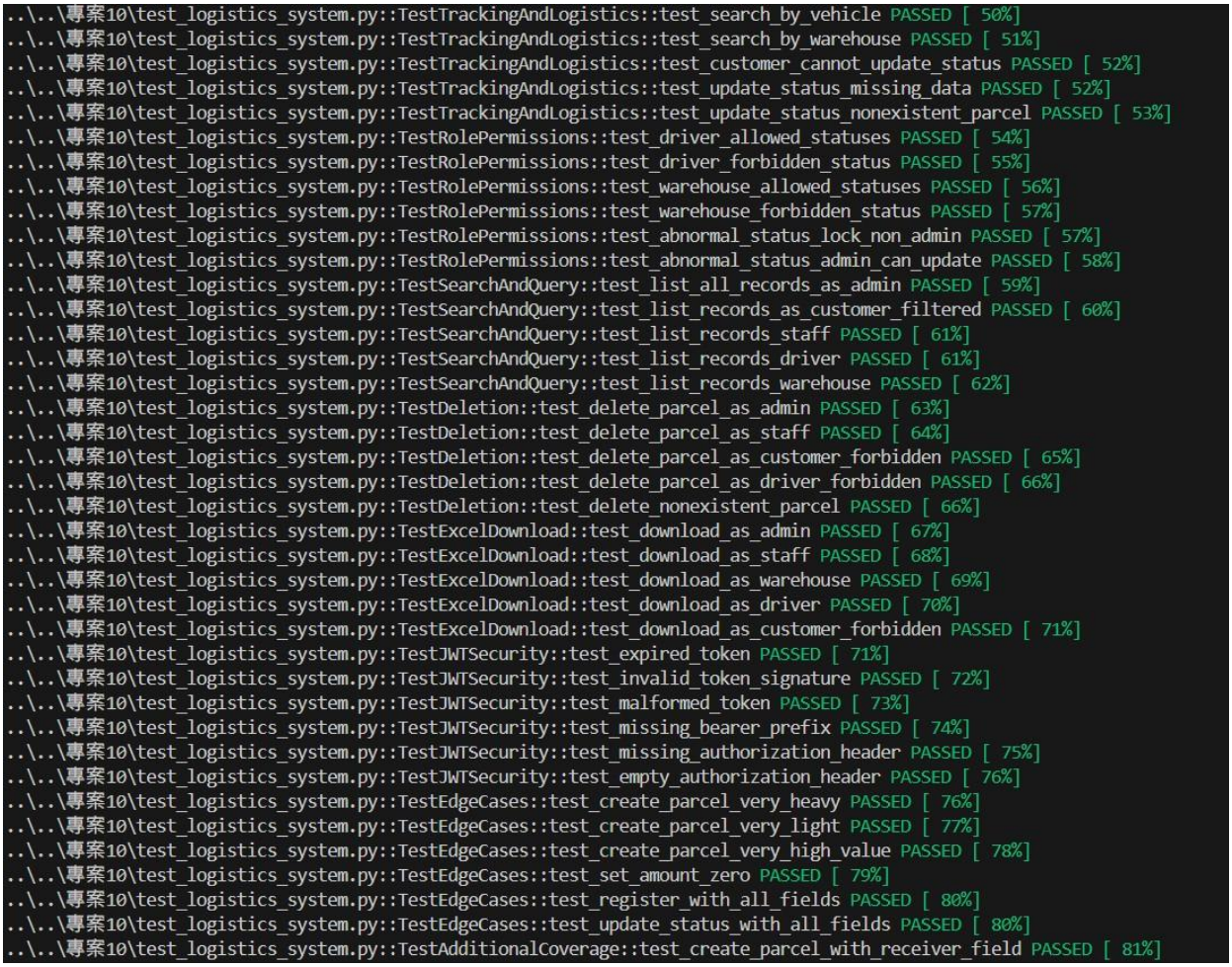
測試編號 (ID)	測試模組	測試情境說明 (Scenario)	輸入資料 (Input)	預期結果 (Expected Result)	結果
TC-01	**身分驗證**	測試管理員與客戶登入及 Token 生成	帳號: admin, 密碼: correct_pw	回傳 200 OK 並包含有效 JWT Token	**通過**
TC-02	**建立包裹**	測試建立包裹後，是否自動產生追蹤碼	寄件人: Alice, 收件人: Bob	回傳 201 Created 且單號格式為 TRK-YYYYMMDD-XXXX	**通過**
TC-03	**運費計算**	驗證標準速遞費率計算公式	重量: 5kg, 類型: 標準	金額應為 100 + (5 * 60) = 400 元	**通過**
TC-04	**權限控管**	**(安全性)** 測試客戶嘗試修改包裹狀態	Role: Customer, Action: Update	回傳 HTTP 403 Forbidden (權限不足)	**通過**
TC-05	**物流流程**	**(整合測試)** 模擬完整包裹生命週期	建單 -> 入庫 -> 配送 -> 送達	各階段 Status 欄位正確更新，且歷史事件完整寫入	**通過**

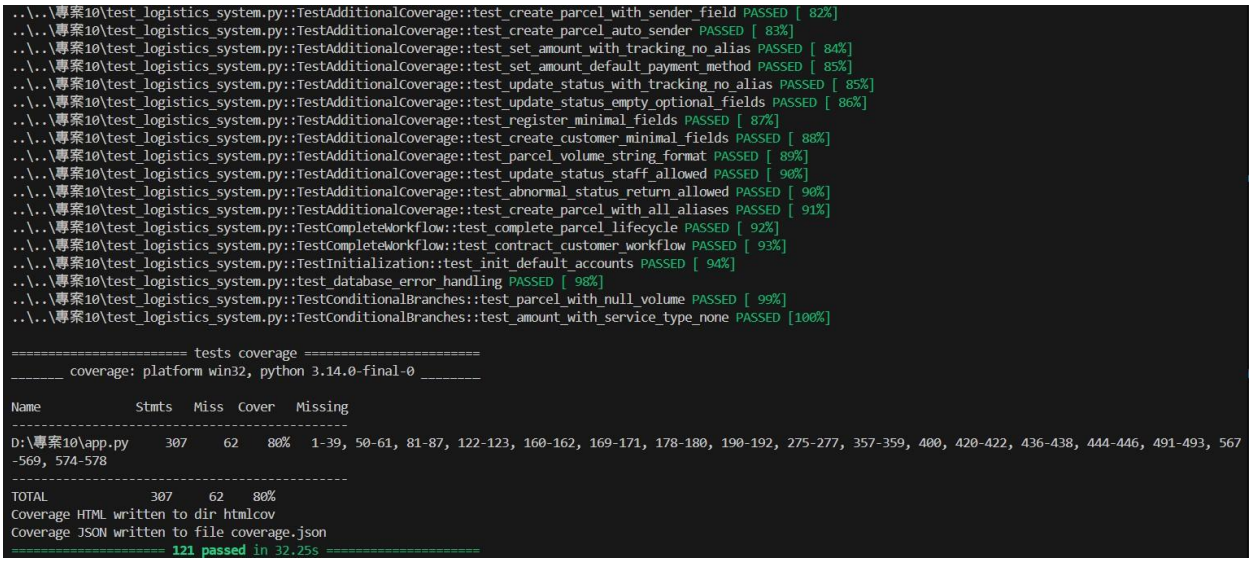
TC-06	**例外處理**	測試資料庫連線失敗或極端輸入	重量: -10kg 或 DB Error	回傳 400 Bad Request 或 500 Internal Server Error (不崩潰)	**通過**

6.4 測試執行結果

```
PS D:\vs code\Microsoft VS Code> & C:\Python314\python.exe d:/專 案10/test_logistics_system.py
✅ 資料庫初始化完成
==================== test session starts ====================
platform win32 -- Python 3.14.0, pytest-9.0.1, pluggy-1.6.0 -- C:\Python314\python.exe
cachedir: .pytest_cache
rootdir: d:\專案10
plugins: anyio-4.11.0, cov-7.0.0, mock-3.15.1
collected 121 items

..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_success_admin PASSED [  0%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_success_staff PASSED [  1%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_success_driver PASSED [  2%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_success_warehouse PASSED [  3%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_success_customer PASSED [  4%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_wrong_password PASSED [  4%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_nonexistent_user PASSED [  5%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_missing_username PASSED [  6%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_missing_password PASSED [  7%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_login_empty_json PASSED [  8%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_register_success PASSED [  9%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_register_contract_customer PASSED [  9%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_register_prepaid_customer PASSED [ 10%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_register_duplicate_username PASSED [ 11%]
..\..\專案10\test_logistics_system.py::TestAuthentication::test_register_missing_credentials PASSED [ 12%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_create_customer_as_staff PASSED [ 13%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_create_customer_as_admin PASSED [ 14%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_create_customer_unauthorized PASSED [ 14%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_list_customers_as_admin PASSED [ 15%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_list_customers_as_staff PASSED [ 16%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_list_customers_as_customer_forbidden PASSED [ 17%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_list_customers_as_driver_forbidden PASSED [ 18%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_update_customer_as_admin PASSED [ 19%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_update_customer_as_staff PASSED [ 19%]
..\..\專案10\test_logistics_system.py::TestCustomerManagement::test_update_customer_unauthorized PASSED [ 20%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_full_details PASSED [ 21%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_small_box PASSED [ 22%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_envelope PASSED [ 23%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_overnight_delivery PASSED [ 23%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_two_day_delivery PASSED [ 24%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_minimal_info PASSED [ 25%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_invalid_weight_negative PASSED [ 26%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_zero_weight PASSED [ 27%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_invalid_weight_string PASSED [ 28%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_negative_volume PASSED [ 28%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_missing_sender PASSED [ 29%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_missing_recipient PASSED [ 30%]
..\..\專案10\test_logistics_system.py::TestParcelCreation::test_create_parcel_unauthorized PASSED [ 31%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_cash_payment PASSED [ 32%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_cod_payment PASSED [ 33%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_monthly_billing PASSED [ 33%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_prepaid PASSED [ 34%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_online_payment PASSED [ 35%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_with_service_type_update PASSED [ 36%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_missing_tracking PASSED [ 37%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_missing_amount PASSED [ 38%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_negative PASSED [ 38%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_invalid_format PASSED [ 39%]
..\..\專案10\test_logistics_system.py::TestBillingAndPayment::test_set_amount_nonexistent_parcel PASSED [ 40%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_received PASSED [ 41%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_warehouse PASSED [ 42%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_loaded PASSED [ 42%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_in_transit PASSED [ 43%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_delivered PASSED [ 44%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_delayed PASSED [ 45%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_lost PASSED [ 46%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_damaged PASSED [ 47%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_update_status_returned PASSED [ 47%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_get_parcel_history PASSED [ 48%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_get_parcel_history_nonexistent PASSED [ 49%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_search_by_vehicle PASSED [ 50%]
..\..\專案10\test_logistics_system.py::TestTrackingAndLogistics::test_search_by_warehouse PASSED [ 51%]
```





---

7. 展示說明（主要功能）

本系統已完成以下核心功能展示：

1. **使用者登入/註冊**：支援 JWT Token 驗證，區分不同角色介面。
2. **線上寄件 (建立包裹)**：前端自動帶入寄件人資訊，後端自動生成單號。
3. **物流追蹤儀表板**：輸入單號即可看見時間軸式的運送歷程。

4. **進階搜尋**：管理員可依照「車輛 ID」或「倉儲 ID」篩選特定包裹。
5. **報表匯出**：支援將包裹清單匯出為 Excel 檔。

---

8. 風險管理

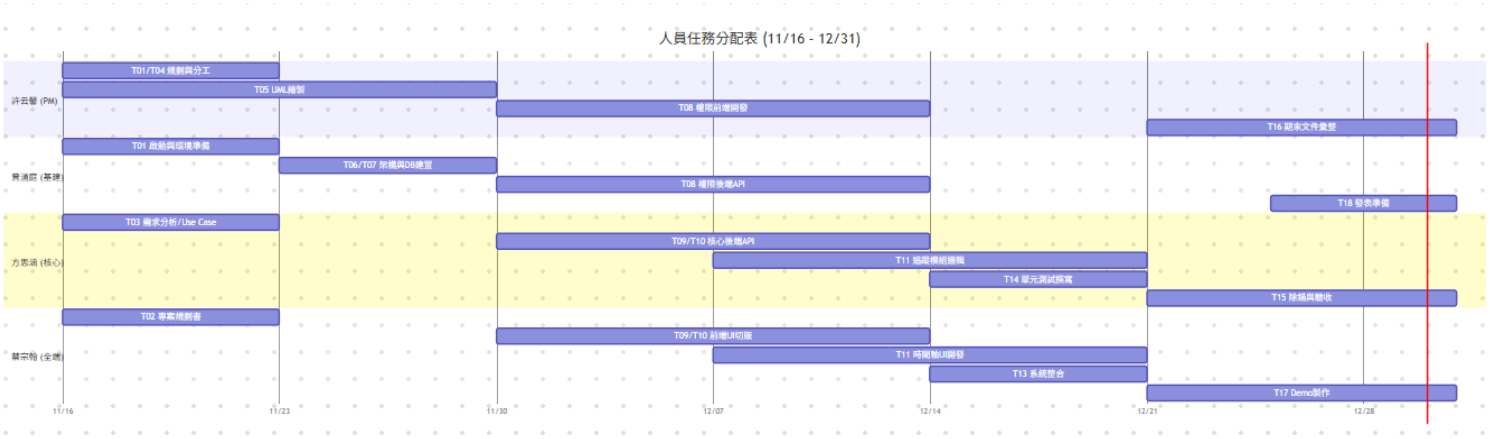
一、一般專案風險

風險	說明	緩解策略
需求變動	客戶方要求額外功能或修改需求	使用變更控制流程，建立需求凍結點
系統整合困難	模組間 API / 介面不一致導致無法整合	在設計階段明確定義 API 規格並撰寫介面文件
技術不熟悉	需導入新框架、新工具，團隊上手時間較長	預留技術學習時間、提供技術 Spike 與原型製作

二、系統運作風險

風險	影響	等級	解決策略
即時追蹤量過大，造成延遲	物流事件無法即時更新，影響查詢體驗	高	使用事件流架構、資料快取
資料一致性問題	狀態不同步、查詢結果錯誤	中	使用事件溯源＋ Read Model 確保一致性
金流連線失敗	無法付款或對帳，造成交易中斷	中	加入重試機制、提供手動對帳流程
角色權限錯誤	客戶可能看到不屬於自己的資料，造成資安風險	高	採 RBAC 角色權限控管、強制驗證、建立審計日誌

9.分工表 (圖)



分工占比:

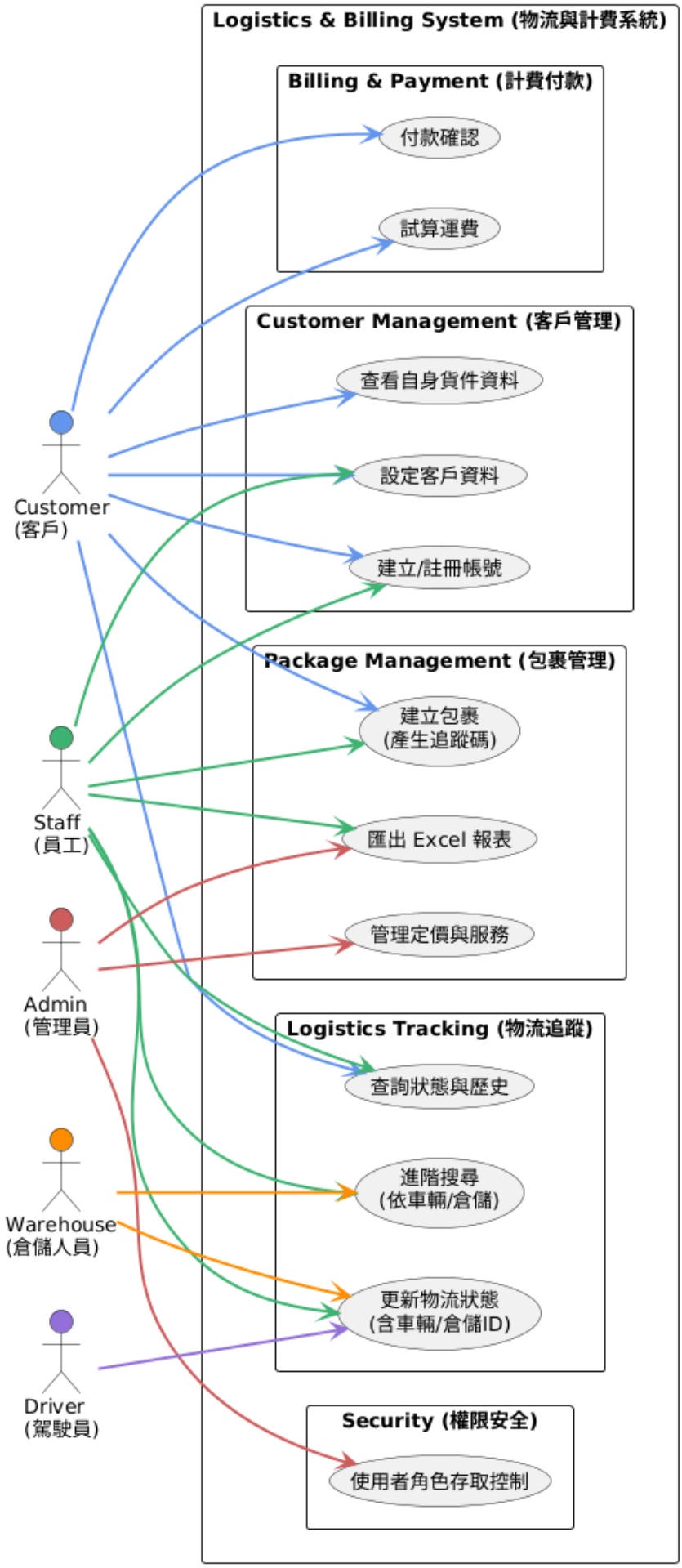
姓名	角色職責	貢獻度佔比	調整理由 (給教授看的潛台詞)
方思涵	核心邏輯與測試	29%	負責最困難的核心演算法與測試除錯，工時最高。
蔡宗翰	前端與金流	29%	負責繁瑣的 UI 切版與整合，工作量大且細碎。
許云馨	專案管理與架構	23%	負責規劃與文件整合，屬於標準 PM 工作量。
曾湧庭	後端與資料庫	19%	負責前期基礎建設，雖然重要但工時集中在前期。
總計		100%	

10.附錄

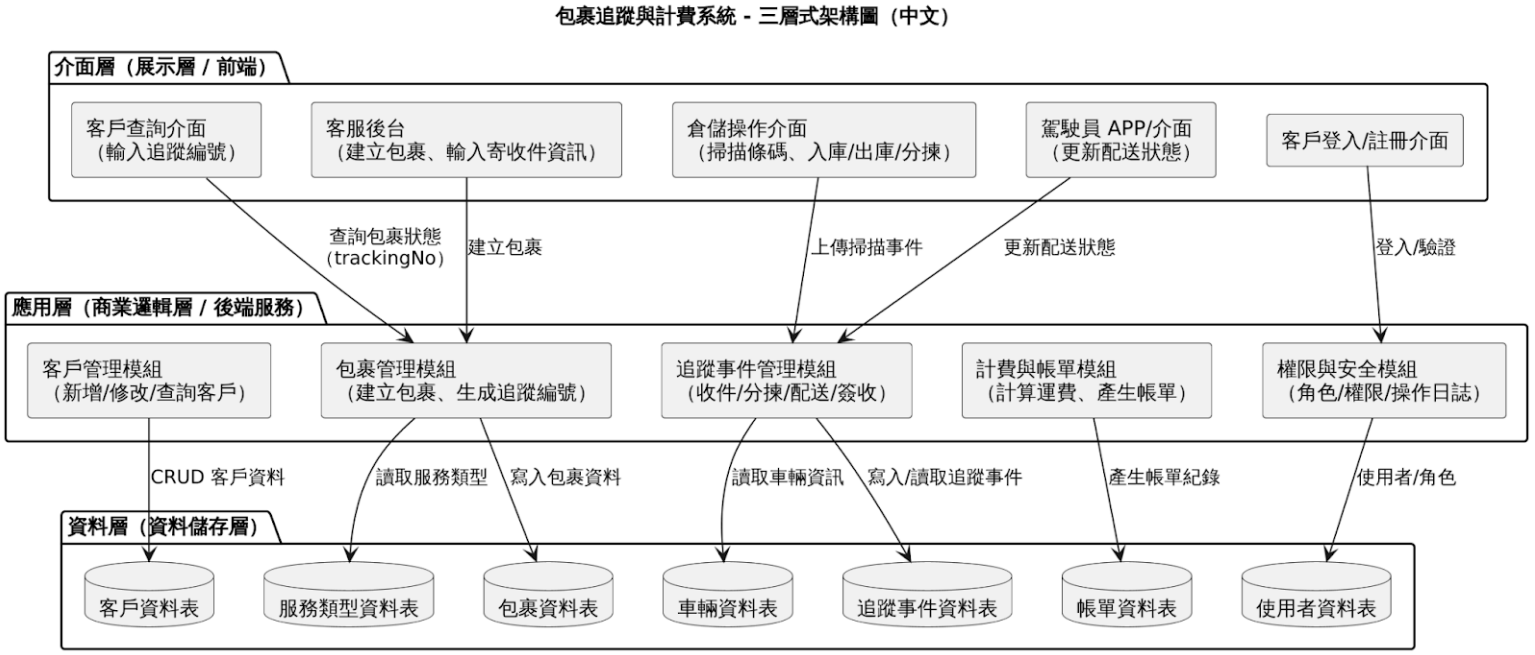
附錄 A：使用案例圖

Please use '!option handwritten true' to enable handwritten

System Use Case Diagram (系統使用案例圖 - 對應程式碼版)

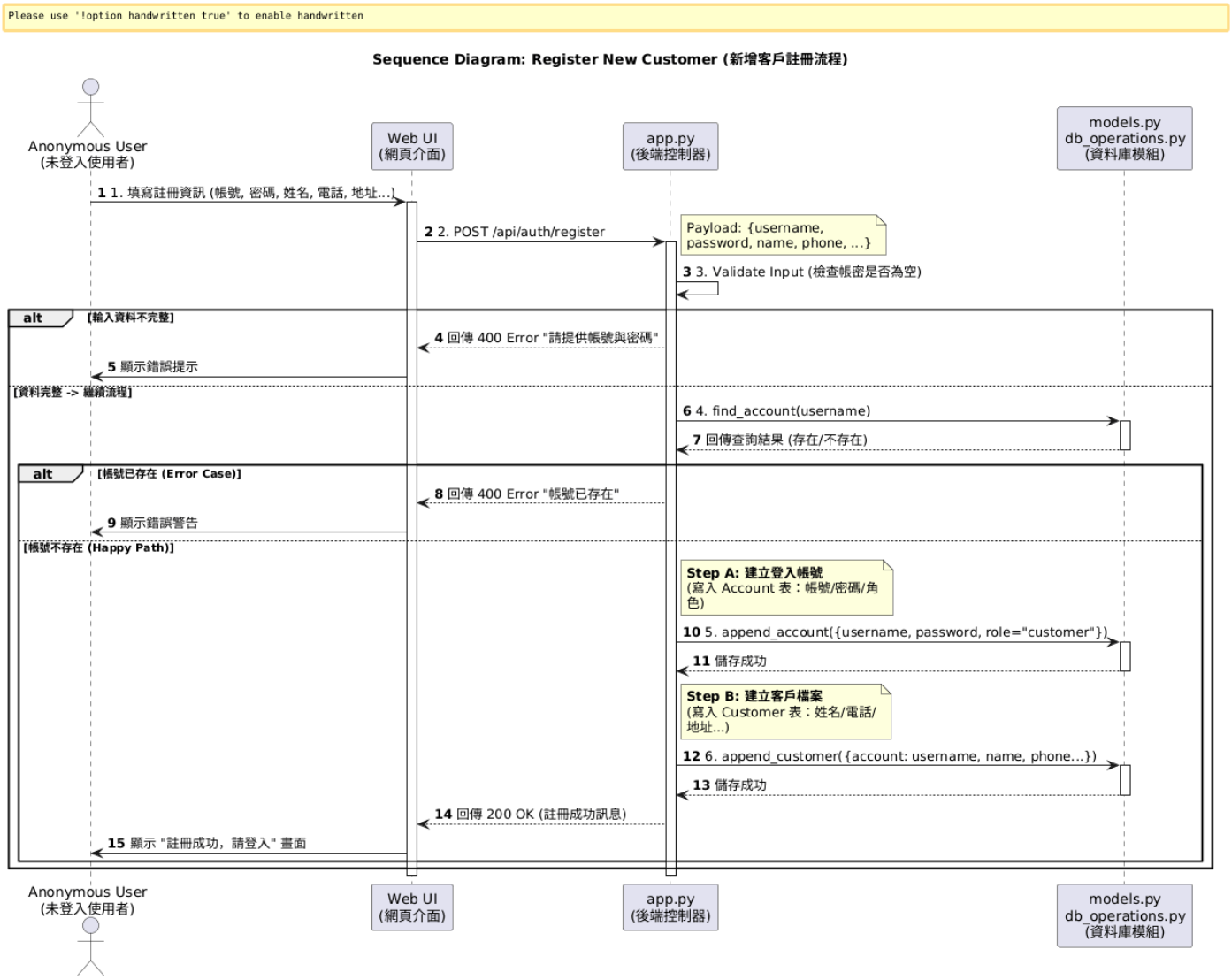


附錄 B：系統架構圖

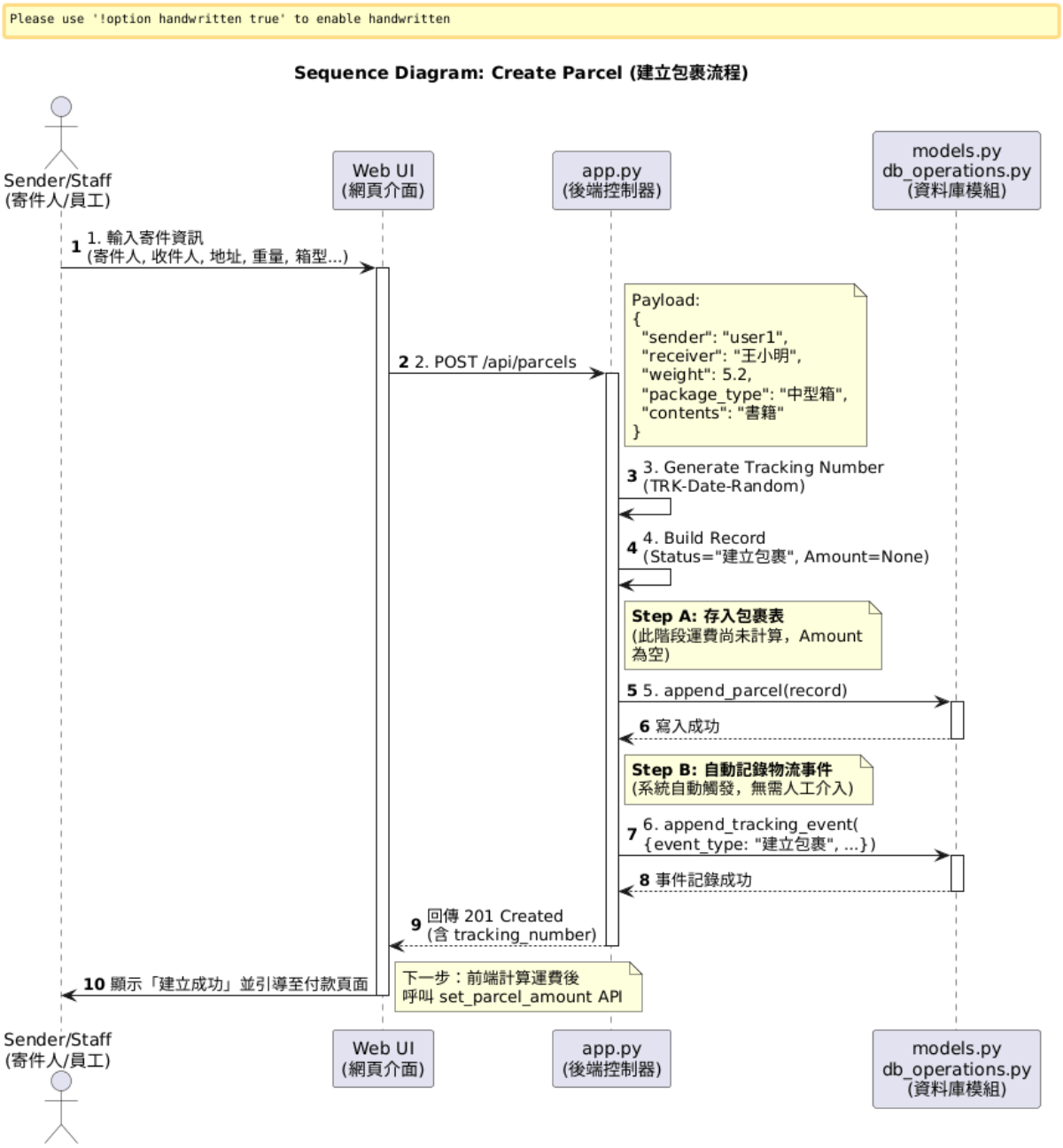

包裹追蹤與計費系統 - 三層式架構圖（中文）
介面層（展示層 / 前端）
客戶查詢介面
（輸入追蹤編號）
客服後台
（建立包裹、輸入寄收件資訊）
倉儲操作介面
（掃描條碼、入庫/出庫/分揀）
駕駛員 APP/介面
（更新配送狀態）
客戶登入/註冊介面
查詢包裹狀態
（trackingNo）
建立包裹
上傳掃描事件
更新配送狀態
登入/驗證
應用層（商業邏輯層 / 後端服務）
客戶管理模組
（新增/修改/查詢客戶）
包裹管理模組
（建立包裹、生成追蹤編號）
追蹤事件管理模組
（收件/分揀/配送/簽收）
計費與帳單模組
（計算運費、產生帳單）
權限與安全模組
（角色/權限/操作日誌）
CRUD 客戶資料
讀取服務類型
寫入包裹資料
讀取車輛資訊
寫入/讀取追蹤事件
產生帳單紀錄
使用者/角色
資料層（資料儲存層）
客戶資料表
服務類型資料表
包裹資料表
車輛資料表
追蹤事件資料表
帳單資料表
使用者資料表

附錄 C：系統序列圖 (Sequence Diagrams)

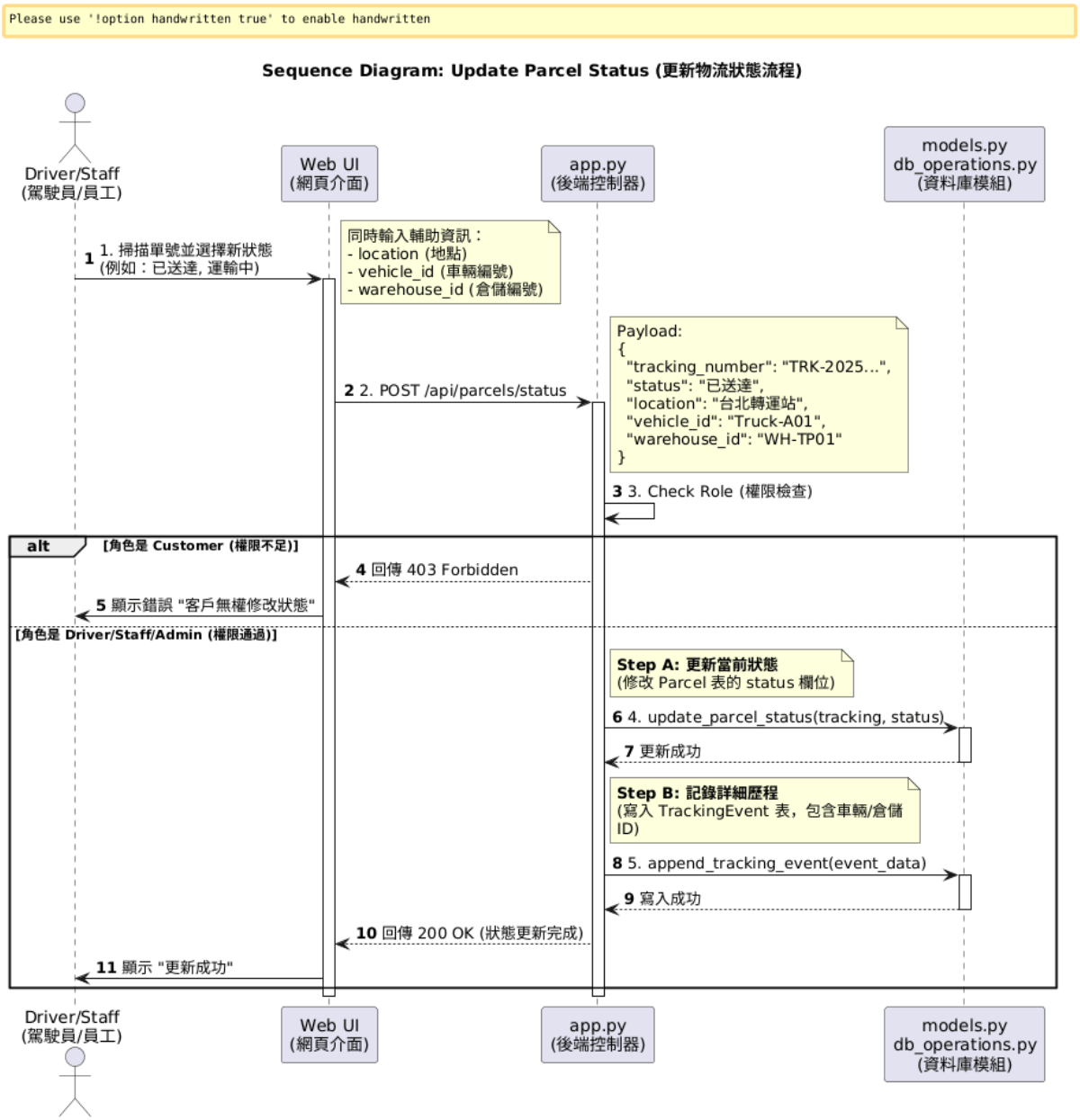
附錄圖 C-1 新增客戶註冊流程



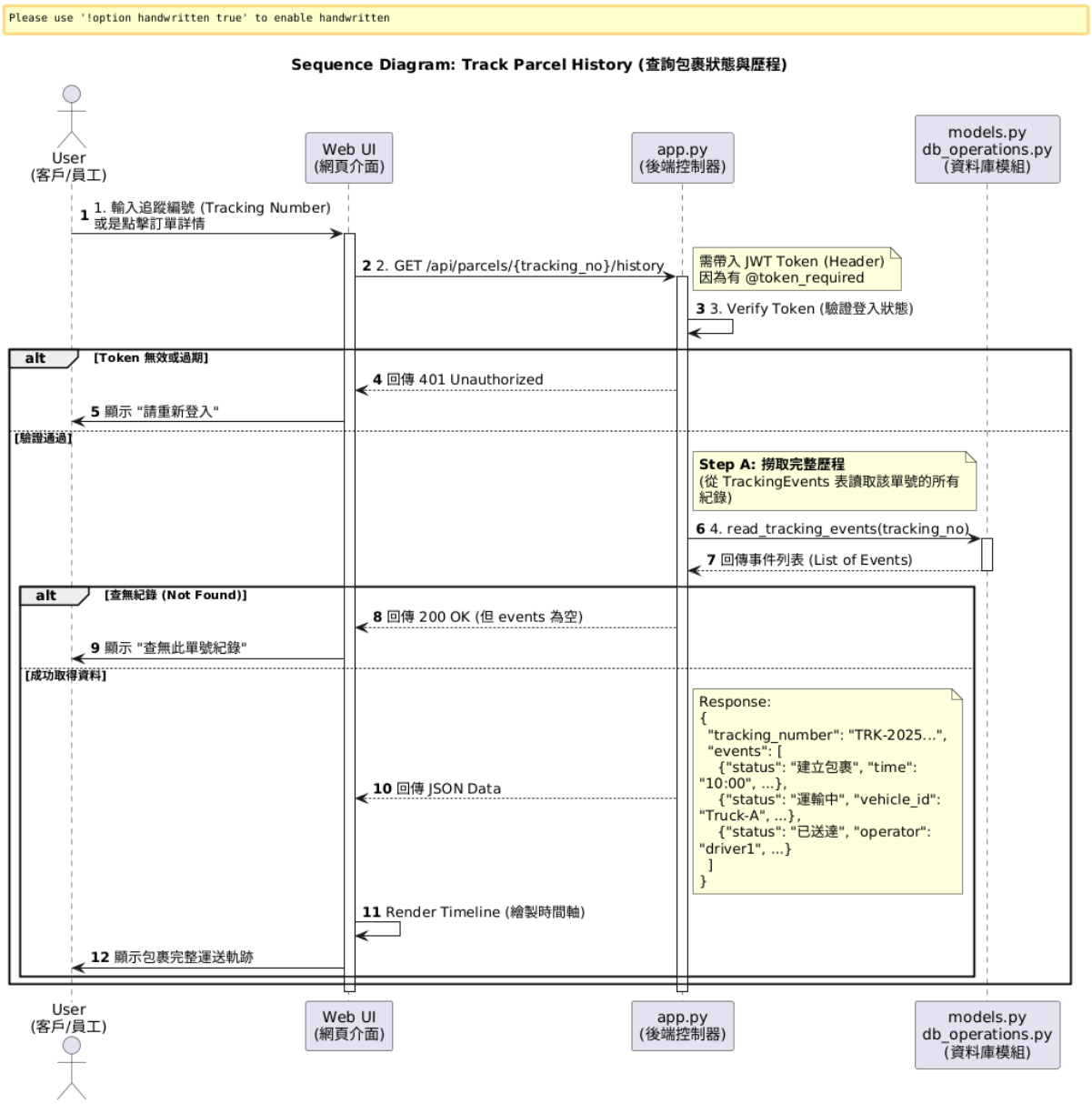
附錄圖 C-2 建立包裹流程



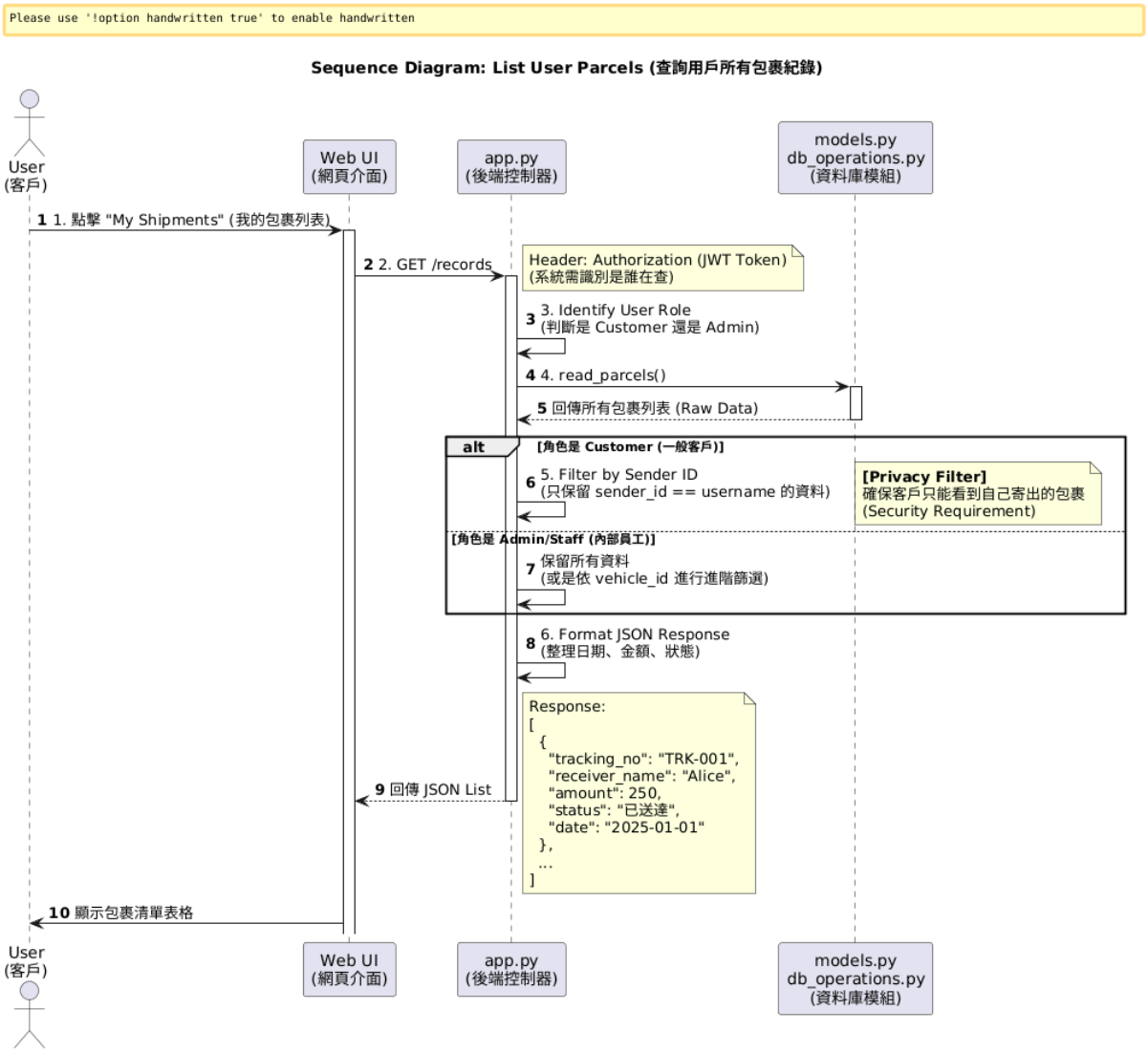
附錄圖 C-3 更新物流狀態流程



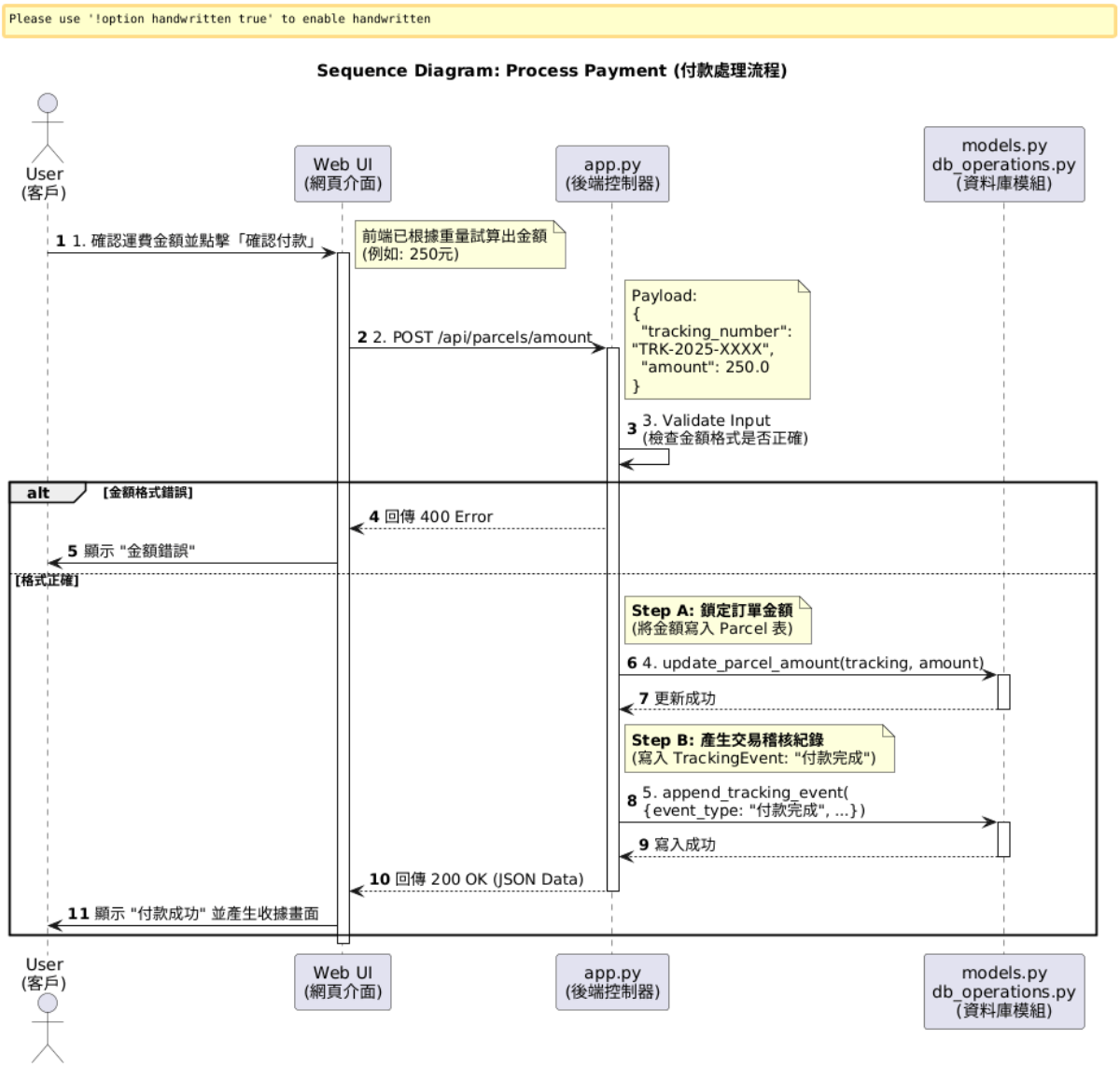
附錄圖 C-4 查詢包裹狀態與歷程



附錄圖 C-5 查詢用戶所有包裹紀錄



附錄圖 C-6 付款處理流程



附錄圖 C-7 權限控制流程