**C#視窗程式設計期末專題報告**

**InventoryManager  
庫存管理系統**

組員：

N10170002 林芳伃

N10170016 孫毓廷

第一章 緒論

■ 研究背景與動機  
中小企業與學校在日常作業中，經常面臨零件、耗材等項目的進出庫管理問題。由於缺乏合適的數位化工具，常以紙本或 Excel 管理，導致資訊落差、庫存錯誤與作業低效。本專題旨在設計一套易於操作且具備紀錄追溯能力的庫存管理系統，以 Windows Forms 為介面，提供直觀的存取方式，並兼顧資料持久化與查詢便利。

■ 研究目的

1. 建置一套具備入庫、出庫、歷史查詢、庫存告警等功能的桌面應用程式。
2. 提供完整操作紀錄，以利稽核與追蹤管理。
3. 界面清晰易懂，方便非資工背景使用者快速上手。

■ 問題陳述  
現行手動記錄或分散檔案管理方式，缺乏即時性與一致性，無法有效處理如下問題：

* 零件資訊更新不同步
* 入出庫變動無法回溯查證
* 缺貨/過庫警示不明確

■ 預期貢獻  
本系統以實作為導向，期望提供：

* 一套可快速部署與實際使用的庫存管理工具
* 清楚的系統結構與可擴充的模組邏輯
* 提供學習資料供後續學生參考或再發展

第二章 系統設計

■ 系統架構圖

1. 前端：Windows Forms GUI（C#）
2. 資料層：JSON 檔案（parts.json, history.json）
3. 控制層：按鈕事件與資料綁定邏輯

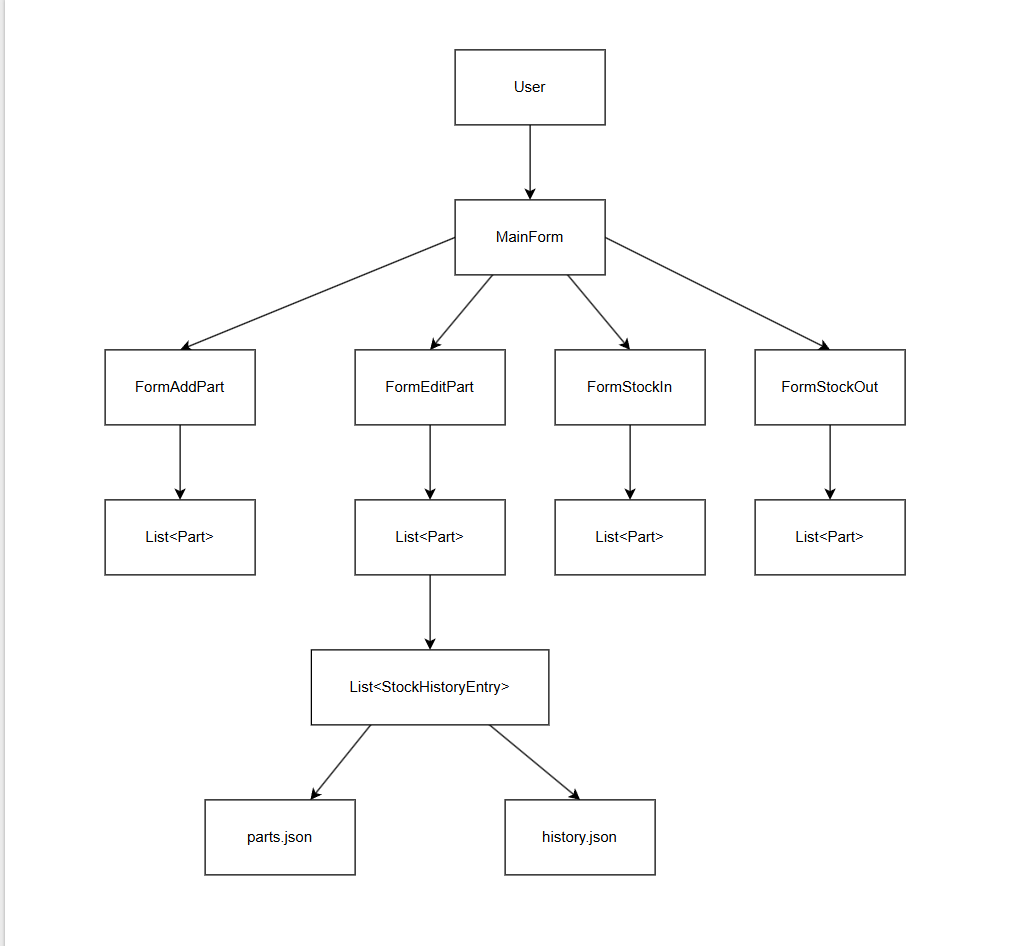


圖 2-1 系統架構圖

■ 功能模組說明

1. 零件管理：新增、編輯、刪除、條件搜尋
2. 庫存操作：入庫、出庫，計算數量變化
3. 歷史紀錄：依時間與關鍵字過濾查詢，追蹤異動
4. 顯示優化：表格欄位自動鎖定與調整、只讀控制

■ 資料庫設計（JSON 結構）

* parts.json：儲存 Part 類別物件清單
* history.json：儲存 StockHistoryEntry 類別物件清單

■ 類別圖 (UML)

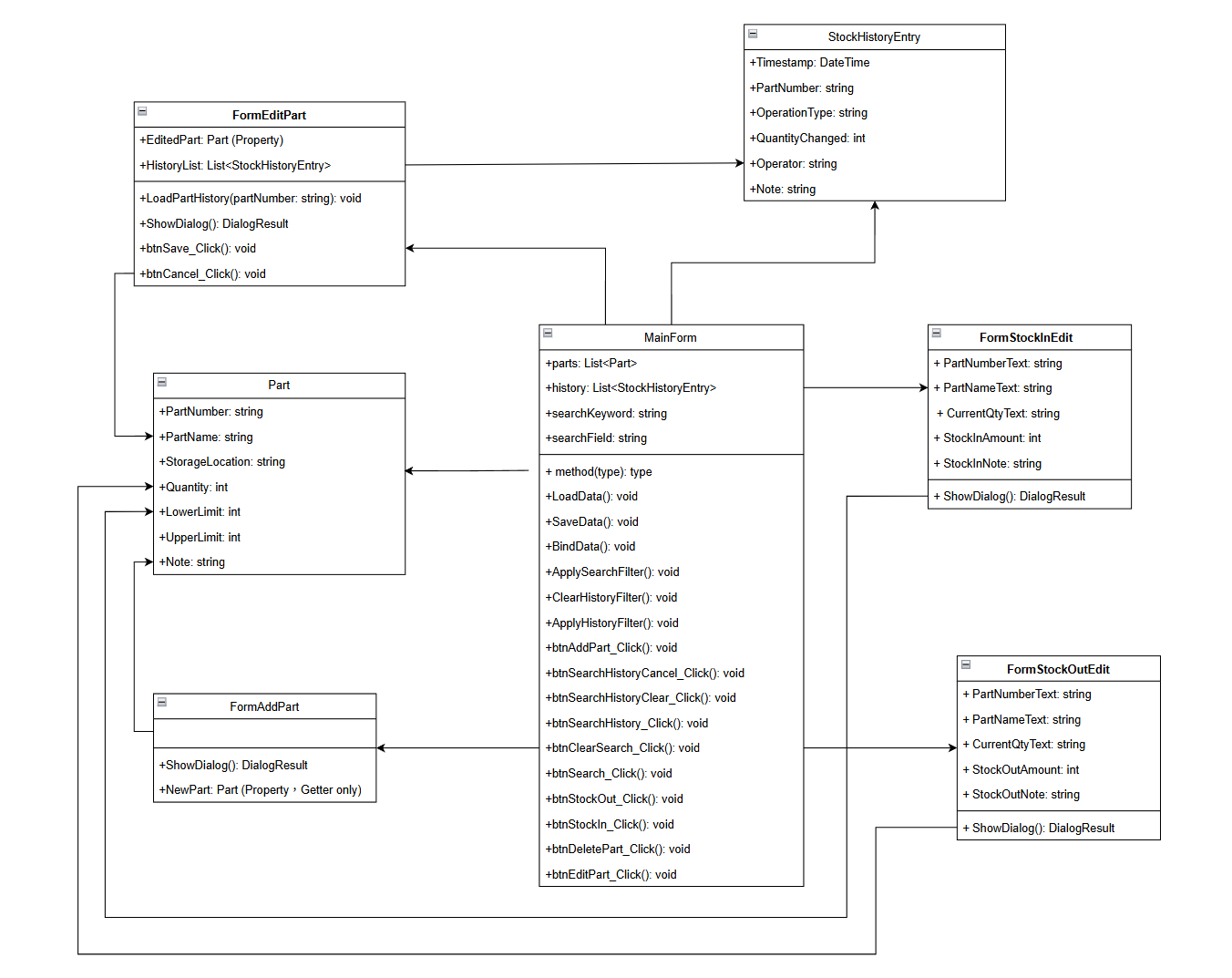
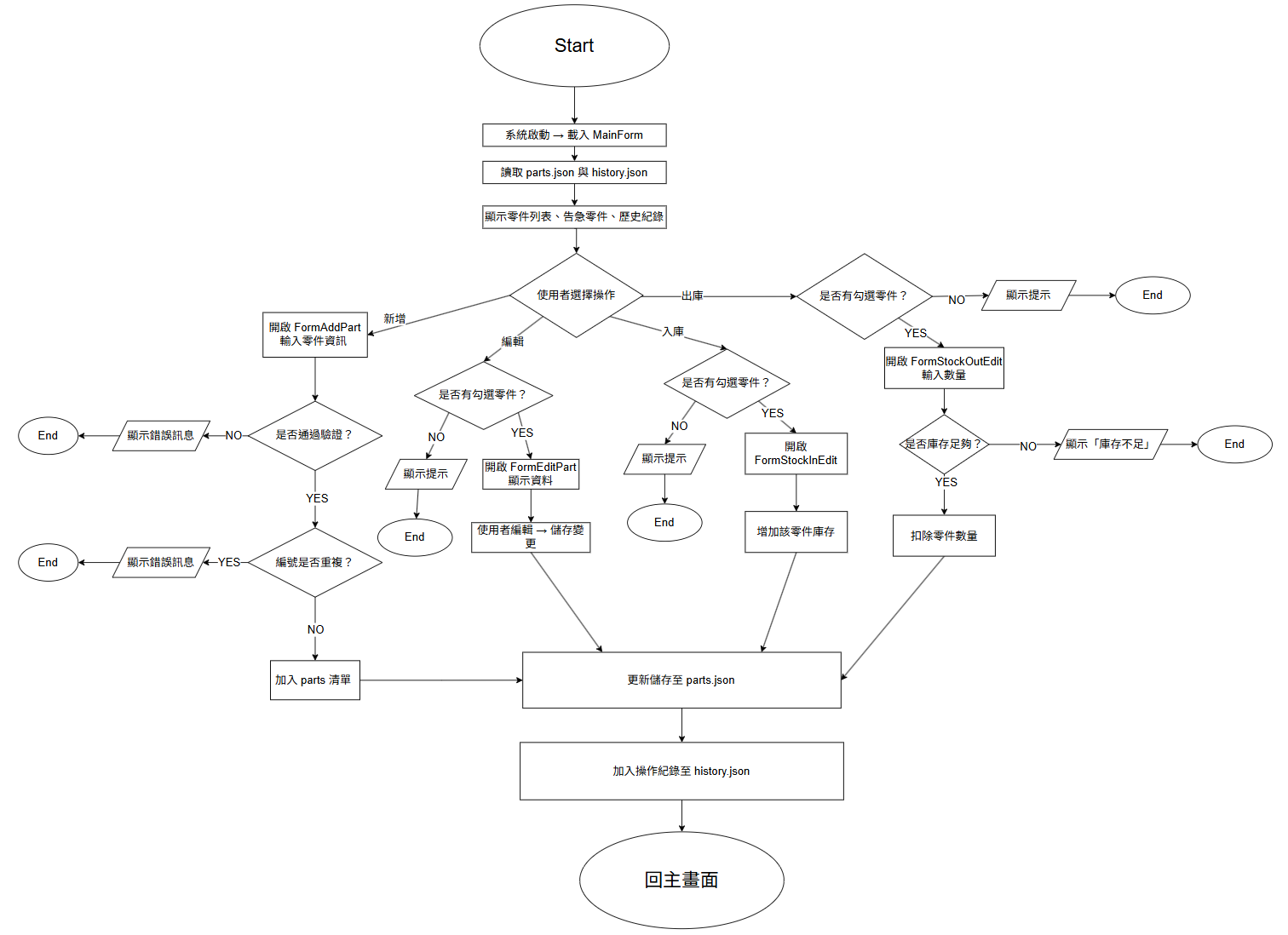


圖 2-2 UML 類別圖（英文）

■ 系統流程描述  
使用者透過視窗介面操作，系統依照以下流程進行處理：

1. 程式啟動：載入 JSON 資料，初始化表格內容
2. 操作流程：
   * [新增零件] 開啟子視窗 → 填寫資料 → 檢查重複 → 加入清單 → 儲存 JSON → 更新畫面
   * [編輯零件] 勾選資料列 → 開啟視窗 → 修改屬性 → 儲存歷史紀錄 → 更新畫面
   * [入庫/出庫] 勾選資料列 → 輸入數量與備註 → 異動數值 → 新增紀錄 → 更新畫面
   * [歷史查詢] 根據時間與關鍵字篩選 → 顯示於資料表格中

  
圖 2-3 系統流程圖

第三章 系統實現

■ 開發環境

* 開發工具：Visual Studio 2022
* 語言平台：C#、.NET Framework 4.8

■ 核心功能代碼解析

* **新增零件功能**



說明：新增後會更新 parts 與 history 並同步儲存資料與更新畫面。

* 編輯零件與變更記錄



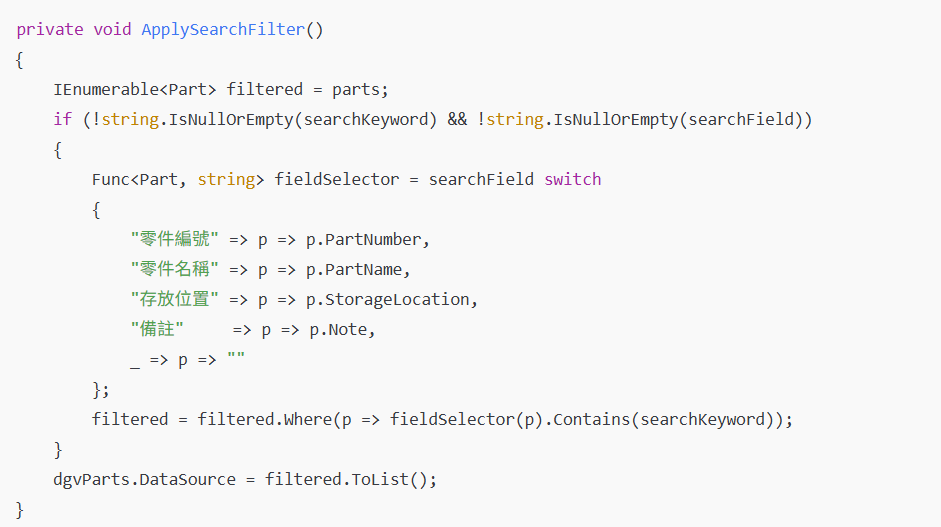
說明：系統會比對修改前後欄位變動，產生完整變更記錄以利查詢。

* 入庫與出庫處理邏輯（以出庫為例）



說明：每次庫存變動都自動寫入歷史，並更新顯示與檔案。

* 關鍵字搜尋（庫存管理頁）



說明：篩選是非破壞性的，即時反映在 DataGridView 上，資料本體不受影響。

**■** 關鍵技術說明

• **資料持久化採用 JSON 結構**  
　透過 System.Text.Json 序列化與反序列化，實現 parts.json 和 history.json 的資料儲存與讀取，不需依賴資料庫即可實現持久化。

• **模組化視窗設計（Form 類別分離）**  
　各功能表單（如 FormAddPart, FormEditPart, FormStockInEdit 等）獨立開發，提升可維護性與重用性，並避免主畫面邏輯過度集中。

• **動態篩選與查詢功能**  
　庫存管理與歷史紀錄皆支援欄位選擇與關鍵字篩選，歷史頁面並提供時間範圍查詢，透過 LINQ 實作資料集的即時過濾。

• **歷史變更自動記錄系統**  
　所有操作（新增、編輯、入出庫）皆會自動產生對應的 StockHistoryEntry 並儲存，包含操作時間、人員、數量變動與備註，提供完整溯源能力。

• **資料表只讀綁定與欄寬固定邏輯**  
　DataGridView 在顯示資料後自動設定為唯讀模式，避免使用者直接編輯造成資料錯誤，並自動調整欄寬以保持界面一致性。

• **欄位異動自動比對並產出變更敘述**  
　編輯功能會比對前後內容，自動記錄異動欄位與內容差異（如「儲位[櫃1→櫃A]」），提升記錄可讀性與可追蹤性。

• **錯誤操作攔截與友善提示**  
　例如出庫時數量不足、未勾選零件即操作等情況，皆有防呆機制與彈出提示，強化系統穩定性與使用者體驗。

■ 界面設計與操作流程

* 三分頁主介面：庫存總覽 / 庫存管理 / 庫存歷史紀錄

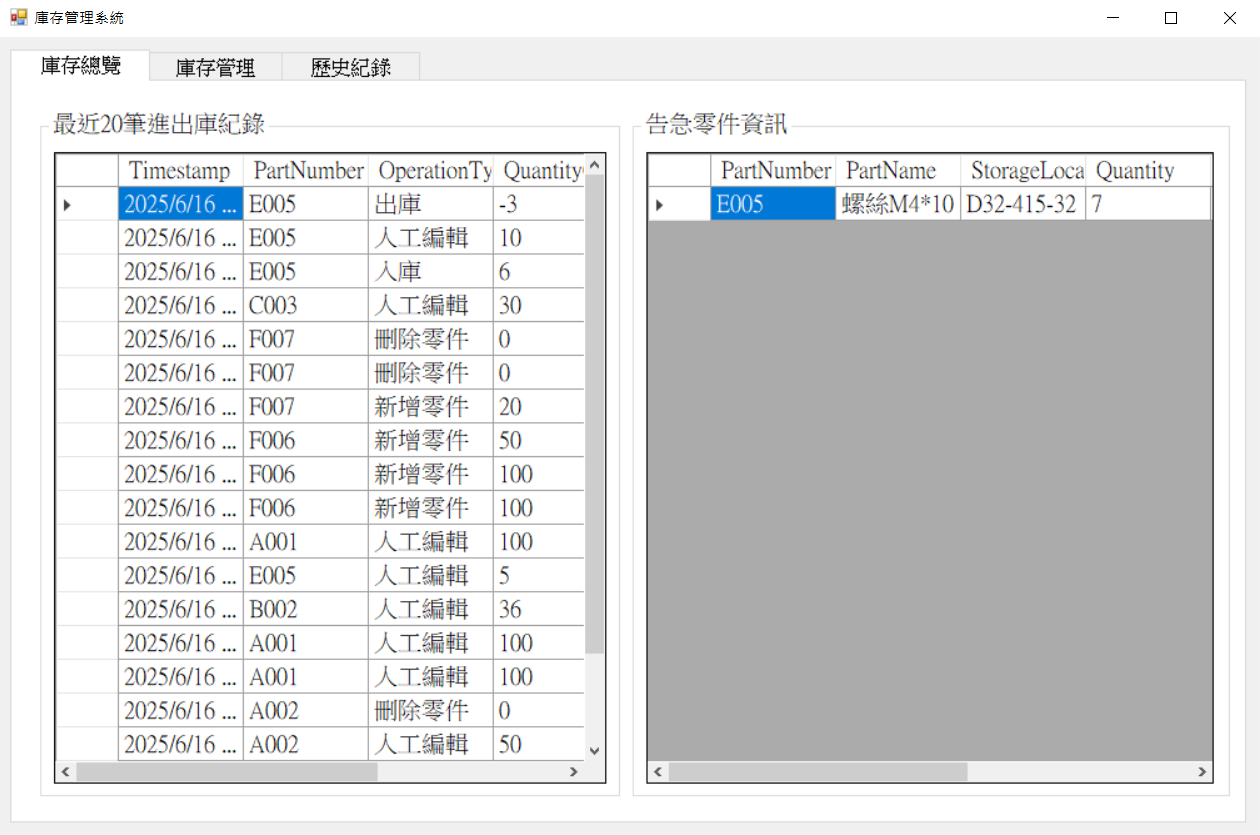


圖 3-1 庫存總覽分頁



圖 3-2 庫存管理分頁



圖 3-3 庫存歷史紀錄分頁

* 所有操作以按鈕驅動（例如新增、刪除、入庫、出庫、搜尋…等）



圖 3-4 庫存管理分頁操作按鈕分布



圖 3-5 庫存歷史紀錄分頁操作按鈕分布

* 新增與編輯零件皆開啟子視窗

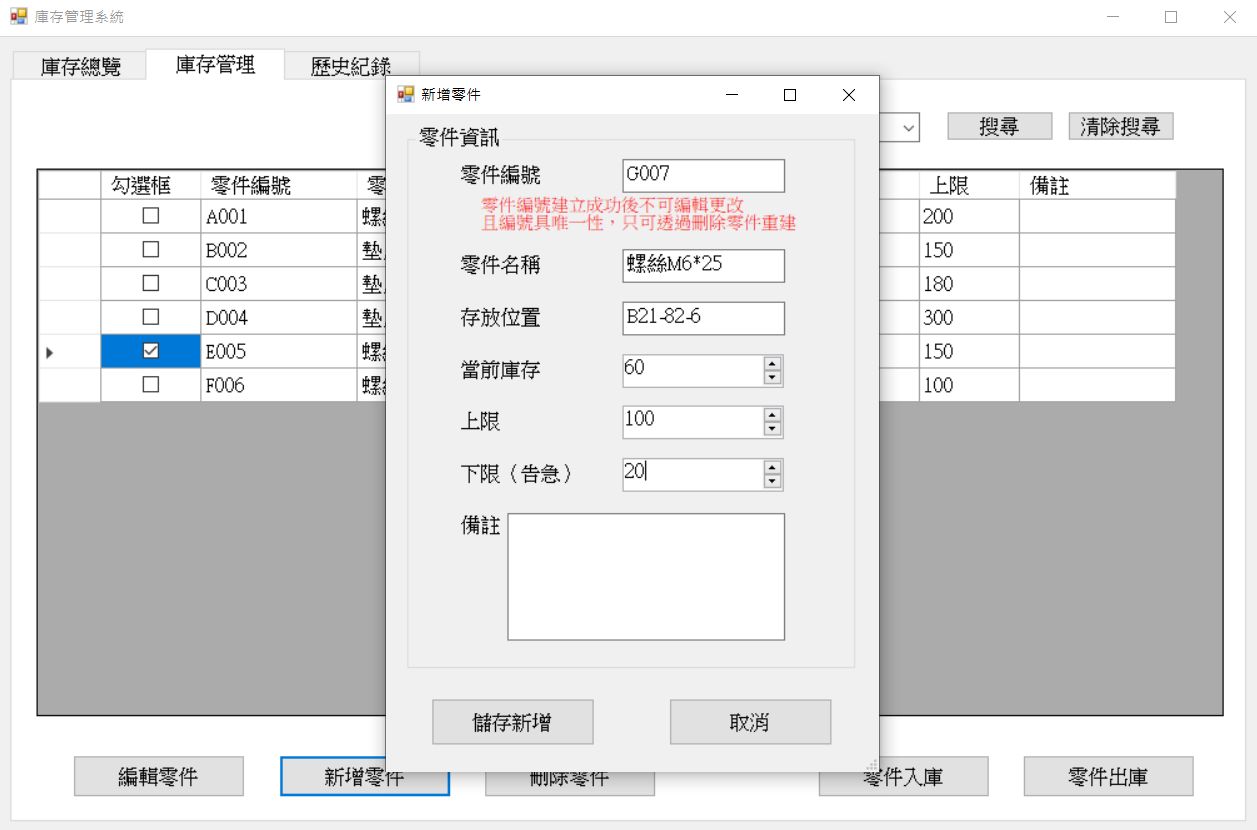


圖 3-6 新增操作視窗



圖 3-7 新增成功提示

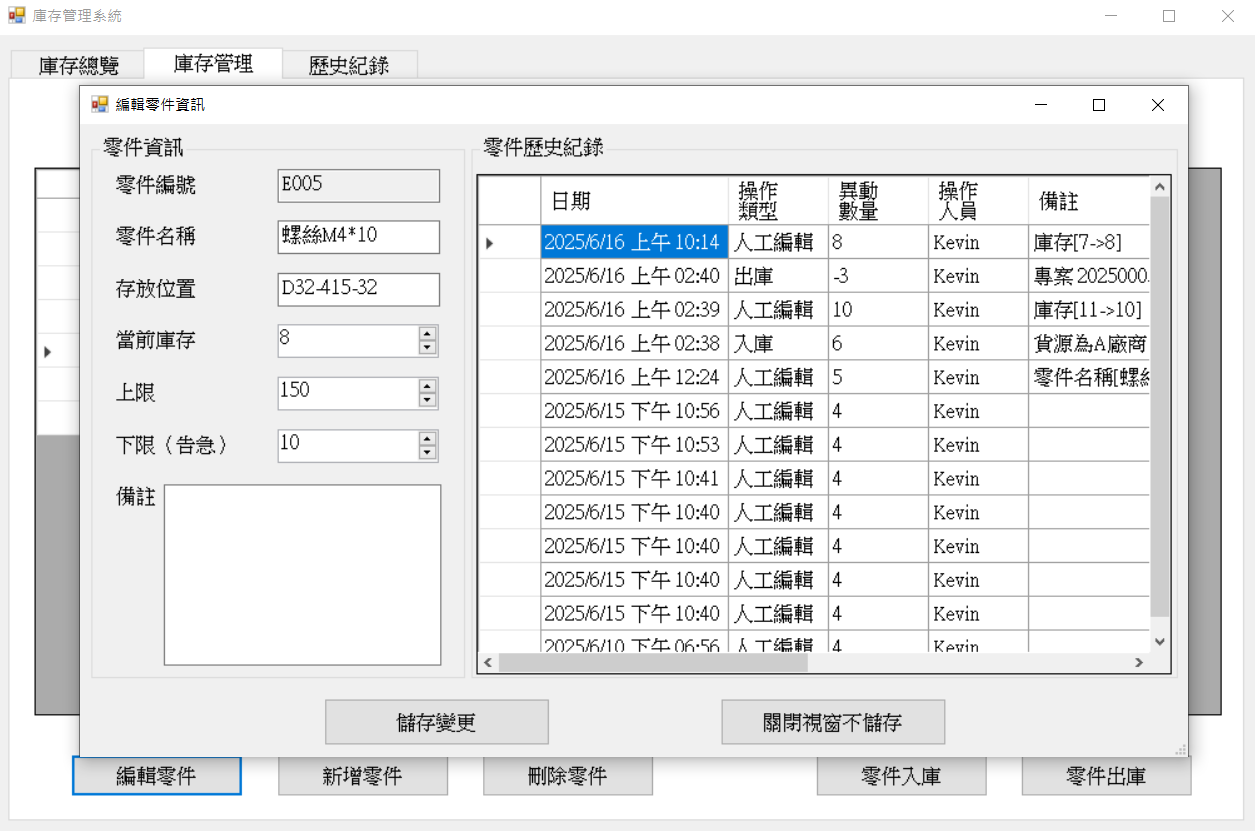


圖 3-8 編輯操作視窗



圖 3-9 編輯成功提示

* 入庫與出庫操作皆開啟子視窗

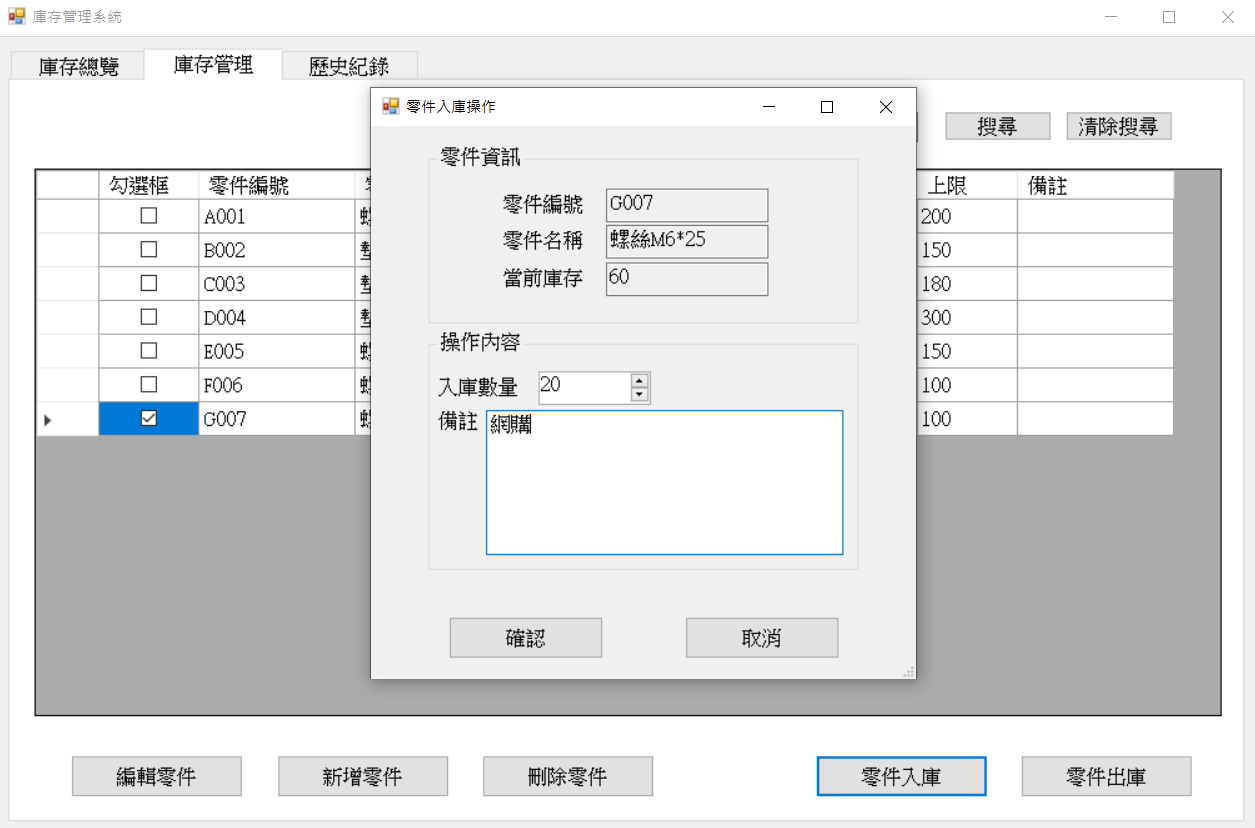


圖 3-10 入庫操作視窗



圖 3-11 入庫成功提示

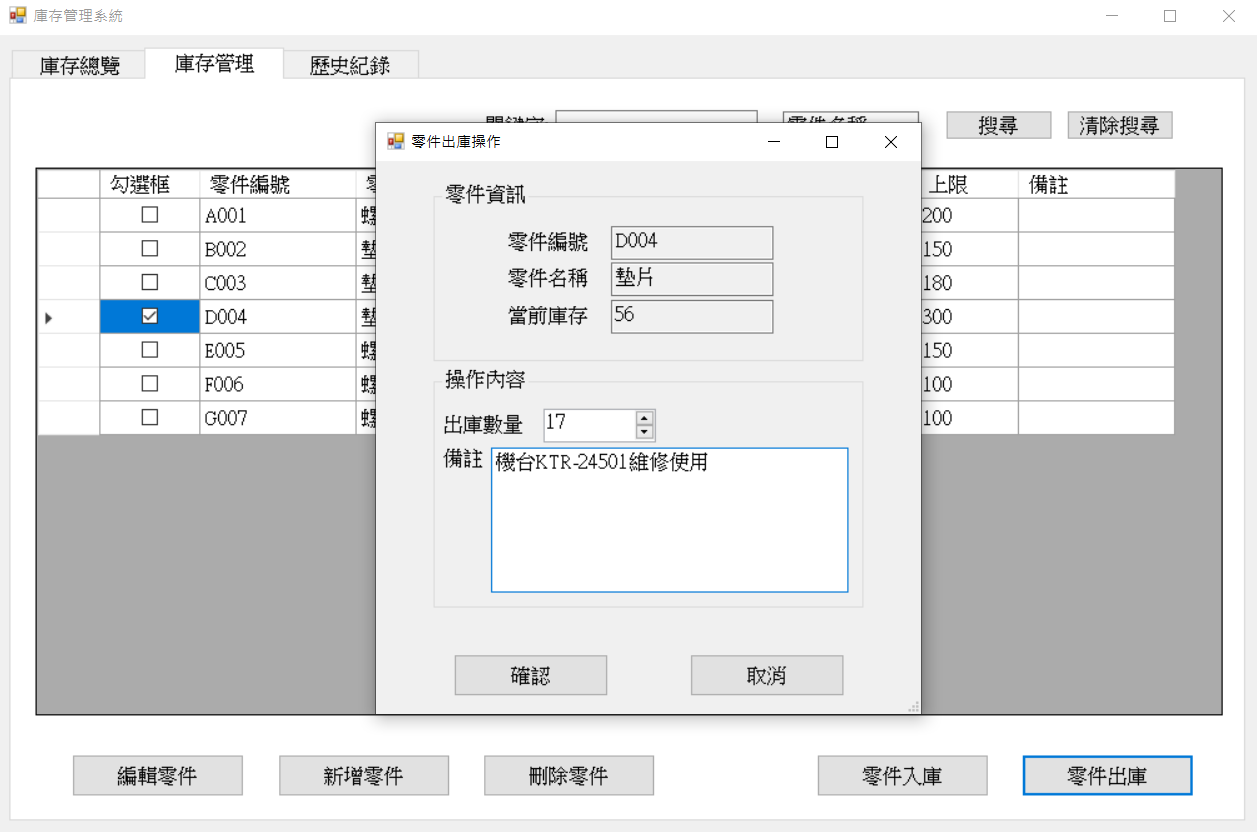


圖 3-12 出庫操作視窗



圖 3-13 出庫成功提示

* 庫存管理可篩選關鍵字範圍



圖 3-14 庫存管理篩選類別選擇



圖 3-15 庫存管理篩選關鍵字

* 歷史查詢可篩選關鍵字與時間範圍

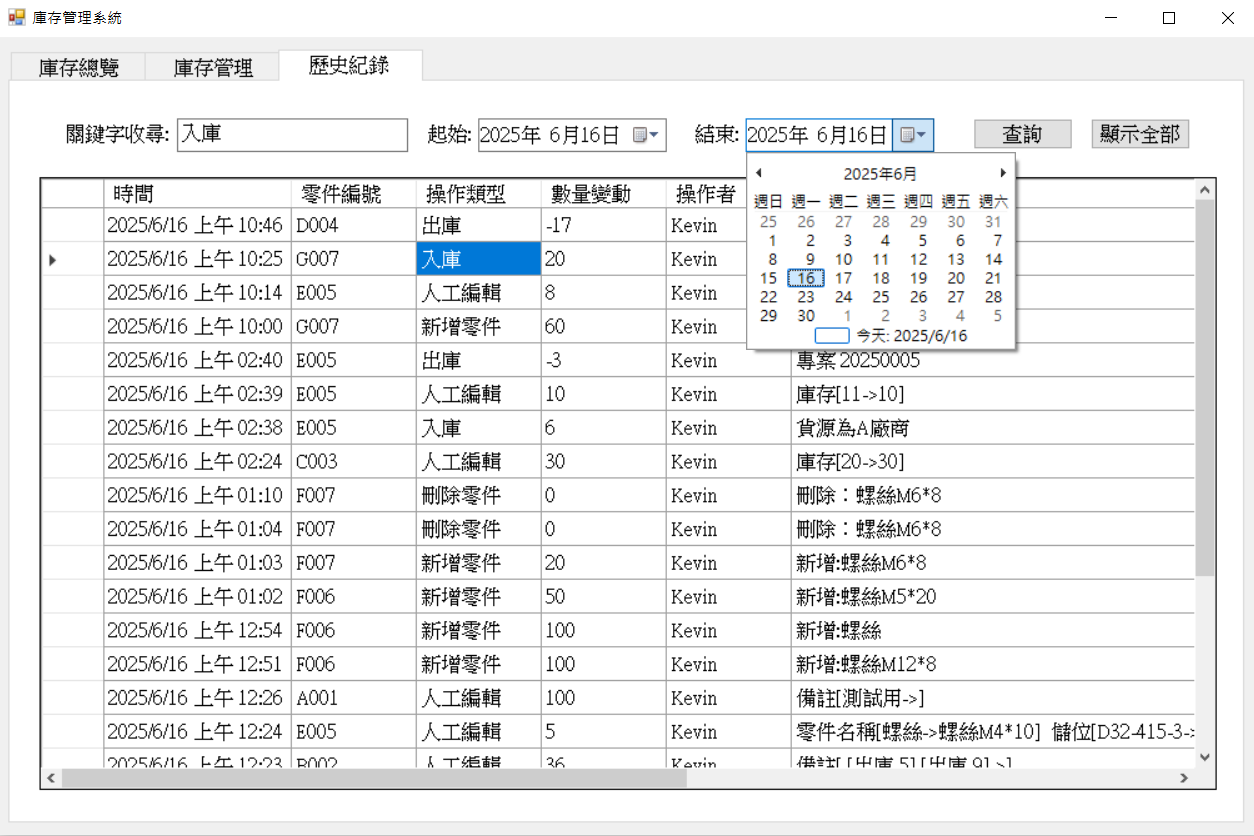


圖 3-16 歷史查詢設定時間範圍

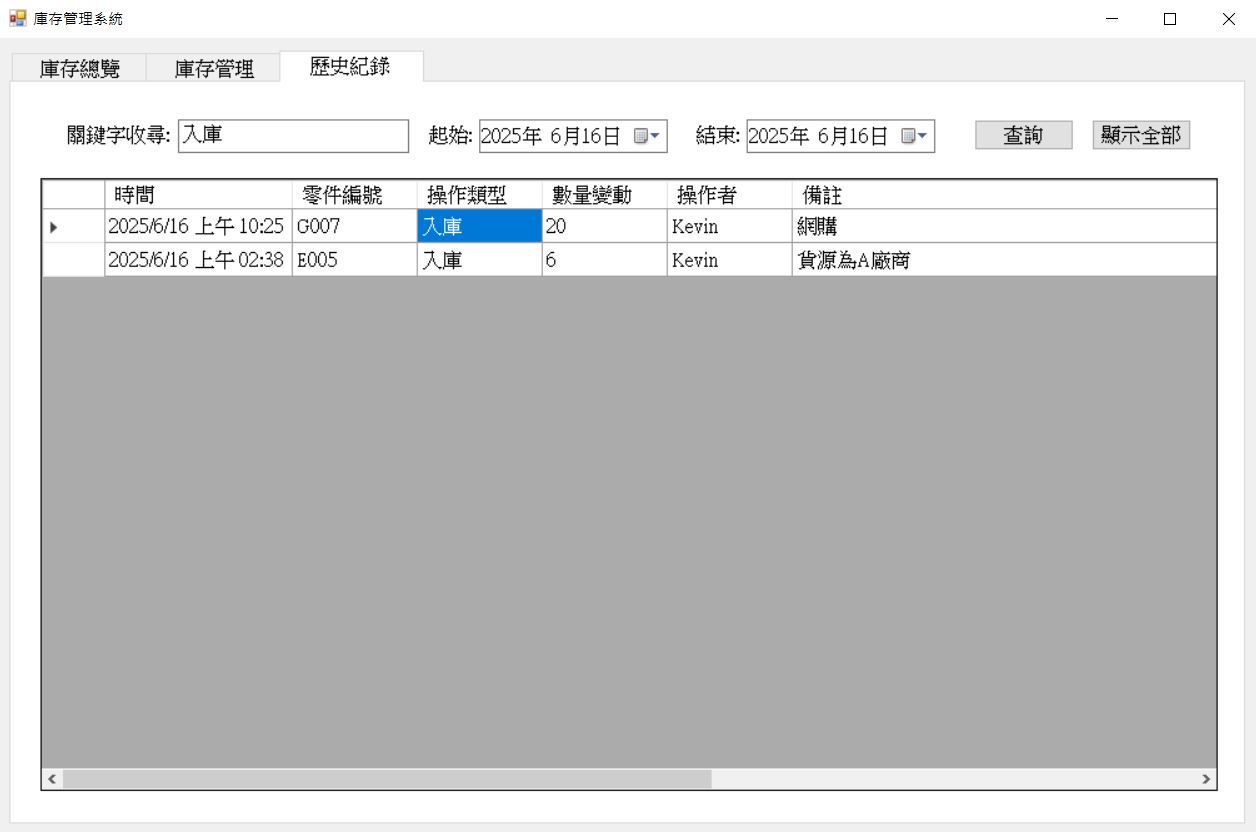


圖 3-17 歷史查詢篩選關鍵字與時間範圍

第四章 測試與驗證

本系統實作完成後，針對各個功能模組進行了單元測試與整合測試，以確保功能正確性與系統穩定性。

■ 單元測試案例

新增零件後，確認 parts.json 正確新增對應資料，並於 history.json 中記錄新增操作歷程。

入庫與出庫功能測試中，確認數量能即時更新至畫面，並於儲存後持續存在於 JSON 檔案中，驗證歷史紀錄紀錄完整性與時間戳的準確性。

■ 系統整合測試

實際模擬完整作業流程：包含零件新增 → 入庫 → 編輯 → 出庫 → 查詢歷史紀錄。

過程中系統能自動刷新顯示區域、正確記錄每次變更、並維持資料一致性與錯誤容忍性。特別針對多使用者操作下的資料一致性做測試，驗證 JSON 檔案寫入時無資料丟失風險。

■ 效能評估

透過操作產生約百筆零件異動資料後啟動程式，確認啟動載入時間穩定於 1 秒以內。

資料綁定、過濾與搜尋操作皆能於瞬間完成，無明顯延遲。即使在高頻繁度操作下，系統仍維持良好響應速度與穩定性，符合中小型倉儲管理系統之實務應用需求。

第五章 結論與未來工作

■ 研究成果總結

本系統成功建構出一套簡潔易用的庫存管理工具，具備資料持久化、操作紀錄、關鍵字篩選與視覺一致性的特色。其模組化的程式結構可利於擴充與維護。

■ 專題限制

* 目前僅支援本地端單一使用者使用，尚未具備網路協同或多人併發處理能力。
* 資料儲存採用 JSON 檔案，對於大量資料與查詢效能仍有提升空間
* 介面固定為 Windows Forms，無法在 macOS 或行動平台使用，限制了部署靈活性。

■ 未來改進方向

1. 可改採 SQLite、MySQL 或雲端資料庫儲存以強化資料一致性與擴充性。
2. 導入登入驗證與權限管理機制，確保紀錄具備操作人身分資訊。
3. 開發簡易版行動應用，供現場工作人員查詢庫存資訊並執行基本操作，提升實務應用廣度與便利性。

附錄. 參考文獻

* Microsoft Docs: DataGridView Control ([https://docs.microsoft.com](https://docs.microsoft.com/))
* Stack Overflow 討論區
* JSON serialization in .NET
* 《C# Windows Form 應用程式開發實戰》