**銀行對帳單自動化工具**

**1. 使用者指南 (User Guide)**

**1.1 工具簡介 (Overview)**

本工具可自動化處理多家銀行的對帳單，將交易資料比對至會計系統的客戶資料庫，並依每天的入帳日期，輸出符合格式的會計憑證檔案。  
它同時會自動避免重複紀錄，並在同一天的多次執行中建立新的檔案（如 YYYYMMDD-2.xlsx, YYYYMMDD-3.xlsx）。

**1.2 安裝步驟 (Installation Steps)**

**Windows 使用者 (Windows Users)**

1. **下載整個專案資料夾** (bank\_reconciliation) 到您的電腦（例如 Downloads 資料夾）。
2. 確保已安裝 **Python 3.10+**：
   * 打開「命令提示字元」，輸入 python --version 檢查。
3. 安裝必要套件（開啟命令提示字元，切換到專案資料夾，輸入）：
4. pip install rapidfuzz pandas openpyxl xlrd pywin32
5. 執行圖形介面：
6. python run\_gui.py

或是到 bank\_reconciliation 中點選 **Run GUI**

**1.3 使用介面說明 (User Interface Guide)**

啟動後會看到主視窗，分為數個區域：

**上方控制列 (Top Controls)**

* **Posting date (YYYYMMDD)**
  + 預設為今天，可手動輸入其他日期。
  + "Today" 按鈕可快速填入今天日期。
* **Add files…**
  + 選取銀行對帳單 Excel 檔案，可一次選多個。
* **Remove selected** / **Clear**
  + 移除選中的檔案或清空全部檔案列表。

**檔案列表 (File List)**

顯示目前待處理的對帳單檔案。

**Run 按鈕 (Run button)**

點擊後，依序處理列表中的檔案：

1. 自動判斷是哪一家銀行。
2. 從銀行對帳單中提取交易資訊。
3. 與會計客戶資料庫比對。
4. 將符合的資料寫入當日的會計憑證檔案：
   * 第一次執行 → YYYYMMDD.xlsx
   * 第二次同日執行 → YYYYMMDD-2.xlsx
   * 以此類推。

**Open output folder**

打開輸出檔案所在的資料夾（通常是 Downloads/Banks）。

**Log 區域**

顯示處理過程，包括：

* 偵測到的銀行
* 成功寫入的筆數
* 重複跳過的筆數
* 輸出檔案名稱

**1.4 輸出結果 (Output Results)**

* 輸出的 Excel 檔位於 Downloads/Banks 資料夾。
* 同一日期的多次執行，會依序建立 -2, -3 後綴檔案。
* 每個檔案只包含該次執行新增的資料，避免重複。

### 📂 使用者目錄說明 (User Directory Guide)

銀行對帳單檔案可放在電腦任意位置，因為執行時會由使用者自行選取。  
程式本身建議放在 Downloads/Bank 資料夾下，方便統一管理與日後更新。

**2. 開發者指南 (Developer Guide)**

**2.1 檔案結構 (File Structure)**

bank.py # 核心邏輯：讀取銀行檔案、比對客戶資料庫、寫入輸出檔案

parsers.py # 各銀行專用的資料解析類別 (e.g., CitiParser, CTBCParser)

fuzzy\_matcher.py # 模糊比對名稱與客戶資料

utils.py # 共用工具，例如記錄跳過的項目

run\_gui.py # Tkinter 圖形介面啟動入口

**2.2 處理流程 (Processing Flow)**

1. **GUI (run\_gui.py)**
   * 收集使用者選擇的檔案與日期
   * 呼叫 bank.py 以命令列方式處理每一檔案
2. **銀行檔案解析 (parsers.py)**
   * make\_parser() 根據檔名決定使用哪個 parser 類別
   * 每個 parser 提取：
     + 客戶名稱
     + 金額
     + 交易細節
3. **客戶資料比對 (fuzzy\_matcher.py)**
   * 根據名稱與資料庫模糊匹配
   * 可人工確認或輸入 ID
4. **輸出 (bank.py)**
   * next\_versioned\_output\_path() 決定輸出檔案名稱（含 -2, -3 後綴）
   * collect\_existing\_counts() 聚合當日所有舊檔的紀錄
   * write\_output() 寫入新檔案，僅保留新的紀錄

**2.3 關鍵邏輯變更 (Key Logic Change)**

* 舊版：同一天的多次執行 → 資料追加到同一檔案
* 新版：同一天的多次執行 → 每次新建 -2, -3 檔案，只包含新增資料  
  → 減少混淆並保留批次處理紀錄

**2.4 開發注意事項 (Dev Notes)**

* **Template file**: TEMPLATE\_FILE 指向空白的會計憑證模板
* **Output folder**: 預設為 ~/Downloads/Banks
* **Duplicate key**: (E date, U cust\_id, S amount)，跨檔案檢查
* **Adding new banks**:
  + 實作 parser 類別 → parsers.py
  + 在 PARSER\_REGISTRY 中註冊銀行關鍵字與類別
* **Testing**: 測試需包含同日多批次的情境，確認 -2、-3 檔案正確產生

### 📂 開發者目錄說明 (Developer Directory Guide)

目前程式以 bank\_reconciliation 專案資料夾為核心，建議維持在單一目錄中集中管理。  
未來若需擴充功能（例如新增銀行 parser、整合其他會計模組），可以在 parsers.py 中新增類別並於 PARSER\_REGISTRY 註冊，或建立新的子資料夾專門存放擴充模組，保持結構清晰。

**📄 輸出格式 (Output Format)**

**注意**：所有產生的檔案現在統一輸出為 **Excel 97–2003 格式 (.xls)**。

此更動是為了 **SAP 相容性** — 目前 SAP 上傳程式僅接受舊版 .xls。

* 程式內部會先建立 .xlsx 檔，再自動轉換為 .xls，最後刪除 .xlsx。
* 輸出檔案範例如下：

會計憑證導入模板 - 20250715.xls

會計憑證導入模板 - 20250715-2.xls

...

這些 .xls 檔案可以直接上傳至 SAP，無需人工轉檔。