好的，這是一份根據您上傳的 Python 程式碼所撰寫的系統規格說明書。文件採用 Markdown 格式，旨在清楚說明系統的架構、功能與技術細節，可作為後續開發或維護的 Coding Spec。

台股分析儀 系統規格說明書 (Coding Spec)

1. 簡介

1.1 目的

本文件旨在詳細說明「台股分析儀」應用程式的系統架構、功能需求、外部介面與資料規格。此系統是一個基於 Streamlit 開發的 Web 應用程式，旨在提供使用者多種台股篩選策略、盤中即時排行分析以及個股深度資訊視覺化功能，作為投資決策的輔助工具。

1.2 系統總覽

「台股分析儀」整合了多個網路資料來源，包含公開財經網站的網頁爬蟲以及 FinMind API 的串接。系統提供一個互動式的 Web 介面，使用者可以透過側邊欄的按鈕觸發不同的分析功能，主要功能區域將以資料表格 (DataFrame) 和多樣化的技術/籌碼/營收圖表來呈現分析結果。

1.3 名詞定義

名詞	英文/縮寫	說明
上市	TAI / Listed	指在臺灣證券交易所 (TWSE) 掛牌交易的股票。
上櫃	OTC	指在證券櫃檯買賣中心 (TPEx) 掛牌交易的股票。
技術指標	Technical Indicators	基於歷史價量資料計算出的數值，如均線(MA)、KD值、MACD等。
籌碼集中度	Chip Concentration	分析特定期間內，股票由少數大戶持有的程度。
FinMind API	-	一個提供台灣金融市場數據的 API 服務。

2. 系統架構

2.1 整體架構

本系統採用單體式架構，由一個核心的 Streamlit 應用程式 (`streamlit_app.py`) 作為使用者介面與業務邏輯的調度中心。該主程式會根據使用者操作，調用不同的功能模組來獲取及分析資料。

- **前端與應用程式伺服器**: 使用 `Streamlit` 框架，負責渲染 UI 介面及處理使用者請求。
- **資料獲取層**: 由多個獨立的 Python 模組構成，各自負責從不同的外部網站或 API 抓取資料。
- **資料分析與視覺化層**: 由核心分析模組 (`stock_analyzer.py`) 和資訊繪圖模組 (`stock_information_plot.py`) 組成，負責計算技術指標並生成圖表。

2.2 模組說明

檔案名稱	模組功能說明
`streamlit_app.py`	**[核心]** 應用程式主入口。負責 UI 佈局、狀態管理、事件處理，並調度其他模組完成使用者請求。
`stock_analyzer.py`	**[分析引擎]** 核心分析模組。負責從 FinMind API 獲取個股價量資料，計算多種技術指標 (SMA, KD, MACD, WMA 等)，並使用 `mplfinance` 繪製詳細的7合1技術分析圖。
`stock_information_plot.py`	**[資訊繪圖]** 個股基本面與籌碼面繪圖模組。負責從 FinMind API 獲取月營收資料、爬取大戶持股比例，並將其視覺化為趨勢圖。
`concentration_1day.py`	**[資料源]** 籌碼集中度爬蟲模組。負責從 `asp.peicheng.com.tw` 爬取資料，並提供篩選函式。
`scraper.py`	**[資料源]** Goodinfo 自選條件爬蟲模組。負責爬取 `goodinfo.tw` 網站上符合特定自訂篩選條件的股票列表。
`yahoo_stock.py`	**[資料源]** Yahoo 上市股票排行爬蟲模組。負責爬取 Yahoo Finance 的**上市 (TAI)**盤中漲幅排行。
`yahoo_stock_otc.py`	[資料源] Yahoo 上櫃股票排行爬蟲模組。負責爬取 Yahoo Finance 的**上櫃 (OTC)**盤中漲幅排行。
`start_server.bat`	**[啟動腳本]** Windows 批次檔，用於一鍵啟動 Streamlit 應用程式 。

3. 功能需求

3.1 使用者介面 (UI)

介面基於 `streamlit_app.py` 進行佈局，分為兩大區塊：

1. **側邊欄 (Sidebar)**:
   - 提供所有功能的觸發按鈕，包含「選股策略」、「盤中即時排行」與「個股查詢」。
   - 「個股查詢」功能包含一個文字輸入框和一個提交按鈕。
2. **主內容區 (Main Area)**:
   - 根據側邊欄觸發的動作，動態顯示分析結果。

- 結果以標題、資料表格 (`st.dataframe`)、資訊提示 (`st.info` , `st.success`)、進度條 (`st.progress`) 及圖檔 (`st.image`) 呈現。

3.2 選股策略功能

3.2.1 籌碼集中度選股 (`concentration_1day.py`)

1. **觸發**: 使用者點擊側邊欄 "1日籌碼集中度選股" 按鈕。
2. **資料獲取**: 系統調用 `fetch_stock_concentration_data()` 函式，爬取 `asp.peicheng.com.tw` 的1日籌碼集中度排行頁面。
3. **篩選邏輯**: 調用 `filter_stock_data()` 函式，篩選符合以下所有條件的股票：
   - 5日集中度 > 10日集中度
   - 10日集中度 > 20日集中度
   - 5日集中度 > 0
   - 10日集中度 > 0
   - 10日均量 > 2000 (張)
4. **結果呈現**:
   - 在主內容區顯示符合條件的股票清單 (DataFrame)。
   - 接著，對清單中的每一檔股票，逐一調用 `stock_analyzer.py` 的 `analyze_stock()` 函式生成技術分析圖，並將圖表顯示在表格下方。

3.2.2 我的選股 (Goodinfo) (`scraper.py`)

1. **觸發**: 使用者點擊側邊欄 "我的選股 (Goodinfo)" 按鈕。
2. **資料獲取**: 系統調用 `scrape_goodinfo()` 函式，爬取一個包含複雜篩選條件的 Goodinfo URL。
3. **結果呈現**: 將爬取到的原始資料直接以 DataFrame 形式顯示在主內容區。

3.3 盤中即時排行分析

3.3.1 / 3.3.2 漲幅排行榜 (上市/上櫃) (`yahoo_stock.py` , `yahoo_stock_otc.py`)

1. **觸發**: 使用者點擊側邊欄 "漲幅排行榜 (上市)" 或 "漲幅排行榜 (上櫃)" 按鈕。
2. **資料獲取**:
   - **上市**: 調用 `scrape_yahoo_listed()` 爬取 `tw.stock.yahoo.com/rank/change-up?exchange=TAI` 。
   - **上櫃**: 調用 `scrape_yahoo_otc()` 爬取 `tw.stock.yahoo.com/rank/change-up?exchange=TWO` 。
3. **成交量預估**: 爬蟲模組內部會根據當前盤中時間，調用 `_get_volume_factor()` 函式計算成交量預估因子，並計算出 "預估成交量"。收盤後因子為 1.0。
4. **分析與篩選流程 (`process_ranking_analysis`)**:
   - **初步篩選**: 篩選 `成交價 > 35` 且 `漲跌幅 > 2%` 的股票。
   - **深度分析**: 對通過初篩的每檔股票，調用 `analyze_stock()` 獲取其歷史資料及 `前5日均量` 。
   - **最終篩選**: 篩選出 `預估成交量 > (2 * 前5日均量)` 的股票。
5. **結果呈現**:
   - 顯示符合最終篩選條件的股票摘要表，包含代號、名稱、成交價、漲跌幅、預估量、5日均量及最新KD值等。

- 將每檔最終入選股票的技術分析圖顯示在摘要表下方。

3.4 個股查詢 (`stock_analyzer.py` , `stock_information_plot.py`)

1. **觸發**: 使用者在側邊欄輸入股票代碼或名稱，並點擊 "生成個股分析圖" 按鈕。
2. **代碼解析**: 調用 `get_stock_code()` 函式，將使用者輸入的字串 (無論是代碼或名稱) 轉換為標準的股票代碼。
3. **圖表生成**: 依序調用三個不同的函式生成圖表：
   a. `analyze_stock()` : 生成7合1技術分析圖。
   b. `plot_stock_revenue_trend()` : 生成月營收趨勢圖。
   c. `plot_stock_major_shareholders()` : 生成大戶股權變化圖。
4. **結果呈現**: 在主內容區垂直依序顯示上述三張圖表。

4. 非功能性需求

4.1 效能

- 所有外部網路請求 (`requests.get`) 皆設定了 10-20 秒的超時 (`timeout`) 以防止無窮等待。
- 對於需要逐一分析多檔股票的功能 (如籌碼集中度、漲幅排行)，處理時間會與符合條件的股票數量成正比。介面中使用了進度條向使用者提供即時反饋。
- 圖表生成後會儲存於 `static/` 資料夾下，避免重複生成。

4.2 可靠性與錯誤處理

- 所有模組中的網路請求、資料解析和檔案操作都被 `try...except` 區塊包圍。
- 發生錯誤時，會在後端 console 印出詳細錯誤訊息 (`traceback`)。
- 在前端介面 (`streamlit_app.py`) 中，會使用 `st.error()` 或 `st.warning()` 向使用者顯示友善的錯誤提示 (例如 "無法獲取資料"、"找不到股票") 。

4.3 相依性

本系統依賴以下第三方 Python 函式庫：

- `streamlit`
- `pandas`
- `numpy`
- `requests`
- `beautifulsoup4`
- `twstock`
- `matplotlib`
- `mplfinance`

- `lxml`

4.4 環境設定

- **API Token**: 系統需要 FinMind API Token 才能正常獲取價量與營收資料。此 Token 必須設定在專案目錄下的 `.streamlit/secrets.toml` 檔案中，格式如下：

  ```
  FINMIND_API_TOKEN = "your_actual_api_token_here"
  ```

- **中文字型**: 為了讓 `matplotlib` 能正確顯示中文，程式碼中已設定字型為 `Microsoft JhengHei` (微軟正黑體)。執行環境需安裝此字型或修改程式碼指向其他可用的中文字型。

5. 外部介面

5.1 FinMind API

- **基礎 URL**: `https://api.finmindtrade.com/api/v4/data`
- **授權**: 透過 HTTP Header 的 `Authorization: Bearer <TOKEN>` 進行。
- **使用端點**:
  i. `dataset=TaiwanStockPrice`：獲取個股日成交資訊 (OHLCV)。
  ii. `dataset=TaiwanStockMonthRevenue`：獲取個股月營收資料。

5.2 網頁爬蟲目標

模組	目標網站	爬取內容
`concentration_1day.py`	`http://asp.peicheng.com.tw/...`	籌碼集中度排行
`scraper.py`	`https://goodinfo.tw/tw2/StockList.asp`	自訂篩選條件之股票列表
`yahoo_stock.py`	`https://tw.stock.yahoo.com/rank/change-up`	上市(TAI)盤中漲幅排行
`yahoo_stock_otc.py`	`https://tw.stock.yahoo.com/rank/change-up`	上櫃(OTC)盤中漲幅排行
`stock_information_plot.py`	`https://norway.twsthr.info/StockHolders.aspx`	個股大戶持股比例歷史資料

6. 資料規格

6.1 核心資料結構

系統中主要的資料傳遞格式為 **Pandas DataFrame**。各爬蟲模組需回傳結構化的 DataFrame，供主應用程式進行處理與顯示。

6.2 圖檔規格

- 所有由 `matplotlib` 和 `mplfinance` 生成的圖表均儲存為 **PNG** 格式。
- 圖檔統一儲存於 `static/` 目錄下，檔名規則化以便管理，例如：
  - `stock_analysis_{stock_id}.png`
  - `revenue_{stock_id}.png`
  - `shareholders_{stock_id}.png`

7. 附錄

7.1 系統啟動

在 Windows 環境下，直接執行 `start_server.bat` 批次檔即可啟動 Streamlit Web 伺服器 。該腳本等同於執行以下指令：

```
streamlit run streamlit_app.py
```