## Project Overview

The DMH\_MR\_Tool is a lightweight internal automation utility designed to streamline the manual process of collecting, extracting, and processing Australian financial market data—such as distribution and component details—from external sources like ASX, iShares, BetaShares, Vanguard, or internal repositories.

It helps users identify relevant updates, extract key data from Excel or PDF reports, and input information into the MR (Master Rate) tab of the internal DMH (Data Management Hub) system based on client-specific business rules.

The tool aims to reduce daily processing time (from 2–3 hours to under 30 minutes) and minimize human error, improving both efficiency and data accuracy.

## Technical Architecture

* **Type**: Desktop application built with Python and PySide6.
* **Target Users**: Single-user environment; not designed for concurrent access.
* **Tech Stack**:
  + **UI**: PySide6
  + **Data storage**: SQLite3
  + **Async I/O**: aiohttp, aiosqlite
* **Deployment Strategy**:
  + **Code** is packaged via internal company macro into a .deploy file.
  + .deploy is uploaded to the **GMAS** platform.
  + End users download a .gmas file (~8KB) and run it via Automation Launcher.
  + **Automation Launcher** loads code from the cloud and runs it using a built-in Python interpreter.
* **Performance Considerations**:
  + Small-scale data volume (~10–20 records/day).
  + Local SQLite3 used for temporary storage.
  + Tool runs asynchronously where possible, but overall usage is light and single-threaded.
* **Compatibility Risks**:
  + Automation Launcher uses a predefined Python environment.
  + Dependency versions may differ from local dev setup, occasionally leading to runtime issues after deployment.
  + All the hard code vars (e.g., websites address, database address, local/remote file path etc.) must be set in a configuration file, i.e., “.ini” file, in a shared drive (not packaged in the project).

## Data Specifications

**Data Sources & Formats**

* **ASX**
  + **Announcement Metadata**:
    - Fetched via HTML scraping from:

[Announcement Search by ASX Code and year] https://www.betashares.com.au/fund/high-interest-cash-etf/#announcements

[Daily Announcements] <https://www.asx.com.au/markets/trade-our-cash-market/todays-announcements>

* + - Data includes announcement title, publish date, URL, and associated ASX code(Ticker).
    - Parsed using XPath; stored in SQLite.
  + **Announcement Documents**:
    - Downloaded PDFs (e.g. distribution statements) based on metadata URLs.
* **Vanguard**
  + JSON API returns latest component data for all tickers.
  + Includes: ticker, component values, publish date, ex-date, etc.
  + Internal mapping used to align Vanguard-specific component names with DMH nomenclature.
* **BetaShares**
  + HTML scraped from fund-specific URLs (e.g., [https://www.betashares.com.au/fund/<fund-name>#announcements](https://www.betashares.com.au/fund/%3cfund-name%3e#announcements)).
  + Fund list (~20 funds) cycled through; each returns a list of announcements.
  + Parsed using XPath; similar structure to ASX.
* **iShares**
  + [To be integrated] — source and structure TBD.
* **Processing Logic & Update Frequency**
  + **ASX Daily Feed**: Queried once per day to ingest all new announcements.
  + **ASX by Ticker**: Queried on demand (typically ~10 lookups/day) based on DMH exceptions.
  + **BetaShares/Vanguard**: Queried periodically (~10 times/month) during expected reporting windows.
  + **PDF Downloads**:
    - Triggered when announcements are matched to DMH exception tickers.
    - Stored in a shared network drive and marked as "downloaded" in the announcement database.
    - PDF formats vary significantly across issuers; ASX only hosts documents and does not standardize their structure. The tool supports 5–6 common layout patterns and handles ~10+ edge cases using custom parsers.
* **Data Volume & Performance**
  + **Volume Estimates**:
    - ~100–200 announcement entries/day across all sources.
    - ~1–5 PDFs downloaded on regular days, with peaks of 10–30/day during monthly reporting periods.
  + **Growth Expectations**:
    - Data volume remains low and predictable.
    - No significant scaling concerns due to single-user desktop nature.
* **Storage & Retention Policies**
  + **Metadata (announcements)**:
    - Stored in a SQLite database hosted on a shared network drive.
    - Lightweight structure (ASX code, title, date, URL, download status).
    - Intended for long-term retention with no purge policy.
  + **Documents (PDFs)**:
    - Stored in a shared network drive.
    - Manually cleaned by developers once per year to conserve space.

## Database Schema Details

This tool uses a lightweight SQLite3 database to cache and track downloaded financial announcement metadata and related information. Below are the core tables and their structures.

* **Table** asx\_info

Stores metadata for announcements retrieved from the ASX website based on ASX code or daily queries.

CREATE TABLE if not exists asx\_info (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    asx\_code TEXT NOT NULL,

    title TEXT NOT NULL,

    pub\_date DATE NOT NULL,

    pdf\_url TEXT,

    pdf\_mask\_url TEXT,

    page\_num INTEGER NOT NULL,

    file\_size TEXT NOT NULL,

    downloaded INTEGER DEFAULT 0,     -- 0 = not downloaded, 1 = downloaded, 2 = failed

    parsed INTEGER DEFAULT 0,     -- 0 = not parsed, 1 = successfully parsed, 2 = error occurred while parsing(failed parsed)

    update\_timestamp DATETIME DEFAULT CURRENT\_TIMESTAMP,

    update\_user TEXT NOT NULL

);

CREATE index if not exists info\_code\_date on asx\_info(asx\_code, pub\_date);

* + **asx\_code**: ASX ticker code (e.g., "FLO")
  + **title**: Announcement title
  + **pub\_date**: Publish date of the announcement(yyyy-mm-dd)
  + **pdf\_url**: Direct link to the PDF announcement, can only get by parsing pdf\_mask\_url.
  + **pdf\_mask\_url**: Link to the actual PDF’s url (pdf\_url), stored in the info page html.
  + **page\_num**: Page number of the PDF announcement
  + **file\_size**: File size of the PDF announcement
  + **downloaded**: Flag indicating if the PDF has been downloaded
  + **parsed**: Flag indicating if the PDF has been parsed to business data and stored
  + **update\_timestamp**: Timestamp of when the record was last updated. Defaults to the current system timestamp at insertion
  + **update\_user**: Username of the user who last updated or inserted the record. Required

**Sample rows**:

|  |  |  |
| --- | --- | --- |
| column | Row\_1 | Row\_2 |
| **id** | 1 | 2 |
| **asx\_code** | FLO | VAS |
| **title** | Dividend component details | Distribution Tax Estimates |
| **pub\_date** | 2025-03-11 | 2025-02-07 |
| **pdf\_url** | https://announcements.asx.com.au/asxpdf/20250624/pdf/06l1zp5dnpnylp.pdf | https://announcements.asx.com.au/asxpdf/20250702/pdf/06ld3q9c94dgw9.pdf |
| **pdf\_mask\_url** | https://www.asx.com.au/asx/v2/statistics/displayAnnouncement.do?display=pdf&idsId=02991191 | https://www.asx.com.au/asx/v2/statistics/displayAnnouncement.do?display=pdf&idsId=02989728 |
| **page\_num** | 15 | 2 |
| **file\_size** | 315.8 kb | 171.0 kb |
| **downloaded** | 1 | 0 |
| **parsed** | 2 | 0 |
| **update\_timestamp** | 1753886256 | 1753886264 |
| **update\_user** | Alfred | Colin |

* **Table** asx\_nz\_data

Stores parsed financial data extracted from ASX/NZ announcements. Each row represents a structured data entry linked to an announcement (via **info\_id**) and mapped to a corresponding **asset\_id** in the DMH system.

create table if not exists asx\_nz\_data(

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    asx\_code TEXT,

    info\_id INTEGER NOT NULL,

    pub\_date DATE,

    asset\_id TEXT,

    ex\_date DATE,

    pay\_date DATE,

    currency TEXT,

    income\_rate numeric(8,8),

    aud2nzd numeric(8,8),

    franked\_pct numeric(8,8),

    total numeric(8,8),

    unfranked\_pct numeric(8,8),

unfranked\_cfi numeric(8,8),

    supplementary\_dividend numeric(8,8),

    tax\_rate numeric(8,8),

);

create index if not exists nz\_code\_exdate on asx\_nz\_data(asx\_code, ex\_date);

create index if not exists nz\_id on asx\_nz\_data(info\_id);

* + **asx\_code**: ASX ticker code (e.g., "FLO")
  + **info\_id**: Foreign key referencing announcement metadata (**asx\_info.id**)
  + **pub\_date**: Publish date of the announcement(yyyy-mm-dd)
  + **asset\_id**: Identifier from the DMH system (used for matching exceptions)
  + **Other columns** represent component-level financial data extracted via regular expressions from downloaded PDFs. These are used downstream by business operations and are not expected to be interpreted or modified by developers.
* **Table** vanguard

Stores metadata for announcements retrieved from the ASX website based on ASX code or daily queries.

create table if not exists vanguard\_data(

    id integer primary key autoincrement,

    port\_id text,

    fund\_name text,

    ticker text,

    apir text,

    fund\_currency text,

    as\_of\_date date,

    ex\_dividend\_date date,

    payable\_date date,

    reinvestment\_date date,

    record\_date date,

    cpu text,

    update\_timestamp DATETIME default CURRENT\_TIMESTAMP

    CGCL text,

    --other columns...

    CGDW text,

    INC text

);

create table if not exists vanguard\_mapping(

    id integer primary key autoincrement,

    port\_id text,

    asset\_id text,

    ticker text,

    apir text,

    is\_valid integer,

    update\_timestamp DATETIME default CURRENT\_TIMESTAMP

);

create table if not exists column\_map(

    id integer primary key autoincrement,

    v\_code text, -- component code in vanguard website

    v\_desc text, -- description of this component in vanguard website

    d\_code text, -- component code in DMH backend database system

    d\_desc text, -- description of this component in DMH MR UI

    is\_valid text,

    update\_timestamp DATETIME default CURRENT\_TIMESTAMP

);

create index if not exists map\_port\_id on vanguard\_mapping(port\_id);

create index if not exists data\_port\_id on vanguard\_data(port\_id);

create unique index if not exists data\_ticker\_apir\_re\_date on vanguard\_data(ticker, apir, reinvestment\_date);

create index if not exists map\_ticker\_apir\_re\_date on vanguard\_mapping(asset\_id, ticker, apir);

* **Table** sys\_log

Tracks all major user actions performed in the DMH\_MR\_Tool for monitoring, performance analysis, and debugging.

CREATE TABLE IF NOT EXISTS sys\_log (

    id INTEGER PRIMARY KEY AUTOINCREMENT,

    update\_timestamp DATETIME default CURRENT\_TIMESTAMP,

    user\_id TEXT NOT NULL,

    action TEXT NOT NULL,

    -- Action name, e.g., "launch\_tool", "download\_pdf", "update\_dmh"

    detail TEXT,

    -- JSON/text blob for storing structured metadata (e.g., ticker, file path, status)

    success INTEGER DEFAULT 1,

    -- 1 for success, 0 for failure

    duration\_ms INTEGER

    -- Optional: time taken to complete the action

);

* **Table** parse\_template

Templates used to parse .xlsx or .pdf data in the Parse Interface. Each column name is a component code in DMH backend database system. For parse\_template\_mr, the column names related to d\_code in column\_map table. For parse\_template\_nz, the column names related to column names in table asx\_nz\_data.

create table if not exists parse\_template\_mr(

    id integer primary key autoincrement,

    template\_name text, -- e.g. vanguard\_au, asx\_mit\_notice, perpetual, etc

    INC\_RAT text,

    DOM\_INC text,

    DOM\_DID text,

    -- ... other fields

    FOR\_INC text,

    is\_valid text,

    update\_timestamp DATETIME default CURRENT\_TIMESTAMP

);

create table if not exists parse\_template\_nz(

    id integer primary key autoincrement,

    template\_name text, -- e.g. asx\_dividend, perpetual, etc

ex\_date TEXT,

pay\_date TEXT,

    currency TEXT,

    income\_rate TEXT,

    aud2nzd TEXT,

    franked\_pct TEXT,

    total TEXT,

    unfranked\_pct TEXT,

unfranked\_cfi TEXT,

    supplementary\_dividend TEXT,

    tax\_rate TEXT,

    is\_valid text,

    update\_timestamp DATETIME default CURRENT\_TIMESTAMP

);

## UI System

**Home Interface**:

Software Usage Dashboard/Overview

**Spider Interface:**

Show the latest data update time of each website in database (update\_timestamp column in asx\_info table)

Download the latest day's data information from a custom website (for ASX, user can only download today’s data, and the previous business day’s data)

Download a specific company’s whole year announcements information from ASX website (user should input the ASX code and the Year)

Activate daily spider process (Update the latest information from each website, not included in pdf downloading. Only fetch the daily information, for ASX, it’s the previous business day’s announcements information.)

Note: The pdf file downloading process will not be triggered in this interface, it will be triggered in the “Fetch” button in the MR Update Interface, since it costs a lot of storage, and only needed ones should be downloaded.

A log activities area will show to display the spider’s current workflow.

An info bar would pop out if any errors occurred.

**Parse Interface:**

The Parse Interface allows users to upload and process source documents containing dividend or component data (commonly in PDF or XLSX format).

Users can **drag and drop** a file into the designated drop area (styled in gray, Google-like) and select a parsing template. The parsing template maps source data fields to the DMH backend column definitions (from the column\_map table, using the d\_desc column).

After parsing, the results are displayed in a structured form with the following columns:

Column Name 1 in DB | re pattern 1 | (value after parsed, editable) | comment

Column Name 2 in DB | re pattern 2 | (value after parsed, editable) | comment

Some fields are not parsed directly from the file. Instead, predefined business rules apply (e.g., the sum of multiple fields). These rules are automatically shown in the **Comment** column.

E.g.

Template asx\_mit\_notice:

|  |  |  |  |
| --- | --- | --- | --- |
| Column Name | re pattern | value | comment |
| ex\_date | 2A\\.5\\.s+Ex Date\\s\*\n?(\\d{1,2}/\\d{1,2}/\\d{4} | 2024-09-19 |  |
| income\_rate | 2A\\.9[\\d\\D]\*?(?:AUD|NZD|USD)\\s(.\*?)\\n | 0.4 |  |
| tax\_rate |  | 0.3 | Default value per client specific |

The form always contains all rows defined in column\_map.d\_desc. If a field’s value is None, the row is hidden by default.

Users may add extra rows manually by selecting from a dropdown list of available d\_desc fields and entering values or comments.

Users can edit the re-pattern by double clicking this cell, the value of this row will change in real time.

Users can select one record and click the “delete” button to delete this row. When a row is deleted, the value of this field will be cleared, and the row will be treated as other hidden rows.

If invalid input is detected, an **info bar** is displayed in the top-right corner of the interface to alert users.

**Data Submission Logic with MR Update Interface:**

Once parsing is completed, users may click **Submit**.

A validation process of this template will be called to validate this set of data. If validation failed, an error info bar will raise to alert users to change data. If validation succeeds, process will continue.

A pop-out dialog prompts the user to provide required header identifiers such as:Client\_ID, Asset\_ID, Ex\_Date, Pay\_Date, MR\_Income\_Rate

The **submission package** consists of: 1. Business data payload (parsed values and manually input records; basically it is fully visible data in the UI). 2. Header identifiers (collected in the dialog)

After submission, the data package is transferred to the **MR Update Interface** for integration.

**Additional Notes:**

Users may ignore the drag-and-drop and instead enter a folder path in another text-input field. All files in the folder are then parsed automatically with the **“Hi-Trust UR”** template and submitted to the MR Update Interface in batch mode.

**MR Update Interface:**

The MR Update Interface displays and manages parsed business records before submission to the DMH system. The central area contains a **table widget** with fixed columns:

E.g.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Client\_ID | ASX\_Code | Fund | Asset\_ID | Ex\_Date | Pay\_Date | MR\_Income\_Rate | Type | Actoin\_bar | status |
| AURR | PLUS | REUC | 902XGW000 | 20250731 | 20250806 | 0.89 | Other | Fetch\_button|Parse\_button |  |
| MBFF | FTO | MAR1 | 952LEII3 | 20250701 | 20251031 | 0.004461 | Last Actual | Fetch\_button|Parse\_button |  |
| MBFF | AAA | MAR1 | 999XEL901 | 20250731 | 20251031 | 7.89 | Template - PIII | Fetch\_button|Parse\_button |  |

Each row corresponds to one record. The visible columns show key identifiers (e.g., Client\_ID, Asset\_ID), while the **detailed business data payload** from the Parse Interface is stored in the background and linked to the row.

**Data Synchronization Logic with Parse Interface:**

When a submission package arrives from the Parse Interface, the system checks if a record with the same combination of “Client\_ID + Asset\_ID + Ex\_Date + Pay\_Date + MR\_Income\_Rate” already exists in the table.

If a match is found → the row is updated, replacing its background data.

If no match is found → a new row is created with the identifiers and linked payload.

As a result, the MR Update Interface table always reflects the most up-to-date set of business records.

Real-time submitting status will display in the table’s “Status” column

**User Operations:**

Users can paste bulk data into a **pop-out table window** and save it into the main table.

Records can be **selected (with a tick box)** and either: Submitted to the DMH system (with real-time status updates in the Status column), or deleted from the interface. A “Select All” option is provided.

**Fetch Button**: Attempts to retrieve PDF announcements from the asx\_info table, using identifiers (ASX\_Code + Ex\_Date + Pay\_Date). If found, the system downloads the PDF (Spider Service will get the real pdf\_url from pdf\_mask\_url), parses it(will left “pending parse” in the Status column, then users can double-click it and select the proper template in the Parse Interface), and finally updates the record’s background data.

**Parse Button / Double-Click**: Redirects the user to the Parse Interface to preview or edit the full business data of the record, and re-submit it back if necessary.

Automatically save the backup file (i.e., source flies that received in the Parse Interface, or the file downloaded from the Spider Service after users clicking the “Fetch” button). Save it as a new file in the backup-file path and rename it to {Asset\_ID}\_{Client\_ID}\_{Ex\_Date in %d%b%Y format}\_{EST or ACT} (If the parse template is “Hi-Trust UR”, it will be “ACT”, otherwise it’s “EST”) when updating to DMH successfully (each successfully updated task row will generate one backup file).

Runtime errors trigger a top-right fly-out alert. If errors occur **before submission**, the user can correct inputs and retry. If errors occur **during DMH update**, successfully updated rows are marked **success**, grayed out, and locked, failed or unselected rows remain editable and can be retried.

**DB Browser Interface:**

Users can select a database from a drop-down list (the tables in this program)

Query a remote Sqlite3 database by SQL (Top Area, text field)

Query result will be displayed (Bottom Area, table widget)

Data in the query result table can be re-ordered, exported, selected and copied

**Manual Interface:**

Display the business logic of the MR Update Interface

**Setting Interface:**

Access the software log file in real time

Modify the log file’s path. download file path, backup-file saved path

## Log System:

Log error while executing python functions, display filename, function name, vars, code line when error raised, can be added in front of a function as a decorator (like loguru)

Log files will be saved in a certain path, and renew a file when reached 5mb

If any error occurred, a pop-out window or fly-out window will display to show the error information. For those pre-defined errors, readable information will display. For those unexpected errors, the traceback of python interpreter console’s information will display.

## Access Control:

All users can access the functions of this tool, including SQL queries. But only the developer can execute ADD/UPDATE/DELETE query.

## Test System:

Each business function will have a test unit.