## 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] 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 NOT NULL,

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

    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
  + **downloaded**: Flag indicating if the PDF has been downloaded
  + **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 |
| **downloaded** | 1 | 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),

    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 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

);

## UI System

**Home Interface**:

Software Usage Dashboard/Overview

**Spider Interface:**

Show the latest data update time of each website in database (Card Area)

Download a single day's data from a custom website (Card Area – button)

Download data from a custom website using a custom ASX code (Card Area – button)

Activate daily spider process (Top area - button - Update the latest information from each website and download data required Send the data to backend parser and flow to the MR Update Interface if any data created by parser)

Show the database information, including table, DB address (Card Area)

**Parse Interface:**

Drag and drop a file (usually a .pdf or .xlsx file contains dividend data) in a gray area, then select a template to start a File Parser process and display the result. The UI will display the template in a form first and the display the parsed content in the form.

Column Name 1 in DB | re pattern 1 | (value after parsed, can be edited) | comment

Column Name 2 in DB | re pattern 2 | (value after parsed, can be edited) | comment

Some columns are not directly parsed from the file, a certain number of business rules will apply on them, like it can be the sum of other columns. In this case, the applied business rules will be displayed in the Comment field.

E.g.

Template asx\_nz\_data:

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

This form will include formula and validation column in the future version. As for now, users can submit formula and validation to the developer for adding them in a certain template.

Add an extra row from a dropdown list of column name, and input a value and comment

Invalid input will raise a fly-out-right-top window to alert users.

Submit the entire form (Use added row included) to MR Update Interface. User will be asked to input some extra information used to display in the MR Update Interface in a pop-out window (Bottom area - button)

Or ignore the drop and drop area, input a folder path in a text-input field. The files in the folder will automatically parsed and flow to the MR Update Interface.

**MR Update Interface:**

An editable table area will show, table will have a fixed column name set

E.g.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Client\_ID | Group\_Fund | Fund | Asset\_ID | Ex\_Date | Pay\_Date | MR\_Income | Type | Source File Path | status |
| AURR | LUSC,LUSU | REUC | 902XGW000 | 20250731 | 20250806 | 0.89 | Other |  |  |
| MBFF |  | MAR1 | 952LEII3 | 20250701 | 20251031 | 0.004461 | Last Actual |  |  |
| MBFF |  | MAR1 | 999XEL901 | 20250731 | 20251031 | 7.89 | Template - PIII | C:\Users\999XEL90 – All – 30Jun2025 – EST.pdf |  |

Task data can be pasted in this area, including task info from Parse Interface

Submit this task form to DMH system (bottom Area – Button)

Real-time submitting status will display in the table

Double click a single row in the table will show the specific business data of it, i.e., a pop-out window containing a table with MR columns and values in it, for previewing the data submitted to the DMH system.

Automatically save the backup file, i.e., source file that contains business data and been parsed, whether it’s from MR Update Interface’s Source File Path or already downloaded in the Spider Interface’s daily process, (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}) when updating successfully (each task row will generate a backup file)

Any runtime error will raise a fly-out-right-top window to alert users. If the error occurred before the updating stage, the user could adjust their inputs to fix the error. If the error occurred while updating to DMH, successfully updated entry will show the “success” status, while other remains “failed”.

**DB Browser Interface:**

Query a remote Sqlite3 database by SQL (Top Area)

Query result will be displayed (Bottom Area)

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

**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.