Day Trading Screener – Technical Documentation

# 📌 Overview

This project implements an automated stock screener for intraday trading. It identifies bullish signals using real-time stock data and technical analysis, with a strong focus on modularity, maintainability, and performance.

# ⚙️ Architecture

The application is organized into the following components:

* • `api/`: Fetches data from Financial Modeling Prep (FMP) API.
* • `runner/`: Executes screener logic with batching, parallelism, and cooldown.
* • `db/`: Manages PostgreSQL access, inserts, updates, and caching.
* • `utils/`: Contains filters, logging setup, email handling.
* • `scheduler.py`: Controls time-based scan execution.
* • `run\_full\_scan.py`: Runs a full market scan and updates the watchlist.
* • `watchlist\_scan.py`: Scans only watchlisted tickers.

# 📚 Module Descriptions

screener\_runner.py: Core scanning logic. Executes in batches, applies filters, saves/export results.

watchlist\_scan.py: Triggers `run\_screener()` on watchlisted tickers (from `watchlist\_cache`).

run\_full\_scan.py: Performs full scan and updates `watchlist\_cache` with new bullish tickers.

scheduler.py: Orchestrates time-based tasks for scanning using configurable intervals.

db/writer.py: Handles inserts to `screener\_run`, `stock\_result`, and upserts `watchlist\_cache`.

db/reader.py: Fetches watchlist tickers, run IDs, and screener results.

api/fmp\_client.py: Queries stock metadata, indicators, and fundamentals from FMP.

utils/filters.py: Defines the logic to evaluate whether a ticker is bullish.

utils/exporter.py: Exports results to Excel and triggers email reports.

# 🔄 Workflow

Daily scheduler triggers:

- 09:30: Full scan → fills watchlist\_cache with new bullish tickers  
- 10:30–12:00 and 13:30–15:00: Every 15 min, scan watchlist and update cache  
- 12:00–13:30: Lunch scan (hourly)  
- 15:00–15:45: Optional final full scan

# 🧪 Testing and Debugging

- `run\_full\_scan.py`: Run manually to debug full cycle

- `watchlist\_scan.py --debug`: Run isolated watchlist test

- Logs saved to `logs/scheduler.log` and console for CLI runs

# ✅ Requirements

Python 3.1.1

Libraries:

psycopg2-binary  
python-dotenv  
openpyxl  
pandas  
pytest  
schedule  
pytz  
requests  
tenacity

PostgreSQL database

Google SMPT account - smtp.gmail.com

FMP () Starter account to have an API\_KEY

# 📈 Screener Process Flowchart

Below is a high-level flowchart of the day trading screener process:

1. Scheduler starts →  
2. At 09:30 → Run Full Scan → Save results to DB + Update Watchlist  
3. From 10:30 to 15:00 → Run Watchlist Scan every 15 mins (except 12:00–13:30 hourly)  
4. At 15:00 → Final Scan  
5. Each scan → Screener logic (indicators, filters, save/export/email)  
6. Health check: If no run in 60 min → email warning

# 📌 Bullish Criteria Explanation

The following criteria determine whether a ticker is considered bullish. Each contributes to the `is\_bullish` flag:  
  
1. \*\*Price > VWAP\*\* – indicates strong intraday support  
2. \*\*EMA20 > EMA50\*\* – bullish momentum confirmation  
3. \*\*RSI(14) between 50 and 70\*\* – confirms active buying without being overbought  
4. \*\*Pre-market % change ≥ 1%\*\* (optional tightening)  
5. \*\*Fundamental filters (Beta, Market Cap)\*\* – optionally applied if `--tighten` parameter is passed (default)  
  
The combination of these is evaluated by `is\_bullish()` in `filters.py`. Signal strength is recorded per result.

# 🔁 Full Scan vs Watchlist Scan

- \*\*Full Scan (run\_screener with limit=2500)\*\*: Occurs once at market open. Pulls 2,000+ tickers from FMP screener API.  
 - Saves all screener results  
 - Updates `watchlist\_cache` with new bullish tickers  
 - Triggers Excel/email summary  
  
- \*\*Watchlist Scan (run\_screener with watchlist\_symbols)\*\*: Every 15 mins on subset of stocks in `watchlist\_cache`.  
 - Evaluates current bullishness  
 - Tracks duration (`bullish\_duration`) based on `first\_seen`  
 - Drops expired tickers from the cache

# 📝 Log Files Overview

- `logs/scheduler.log`: Records schedule actions (start, scan types, errors, missed checks)  
- `logs/screener.log`: Captures screener-specific messages (each ticker processed, bullish alerts)  
- `logs/email.log`: Messages sent or failed from the email notification system  
- `logs/backtest.log`: Used during historical model evaluation or ML runs

# 📦 Initial Ticker Universe

At full scan, tickers are pulled from the Financial Modeling Prep API using `fetch\_core\_screener()` with filters:  
- `volumeMoreThan = 500000`  
- `priceMoreThan = 1`  
- `changeMoreThan = 2`  
- `exchange = NASDAQ,NYSE`  
  
Then the top ~2400 symbols are screened.

# 🗃️ Database Schema (9 Tables)

1. \*\*screener\_run\*\* – records each run timestamp (used as `run\_id` FK)  
2. \*\*stock\_result\*\* – core screener results (one per symbol per run)  
3. \*\*watchlist\_cache\*\* – current bullish tickers (tracked daily)  
4. \*\*quote\_cache\*\* – basic price data  
5. \*\*premarket\_cache\*\* – % change before market open  
6. \*\*technical\_indicators\*\* – company name, sector, exchange, etc.  
7. \*\*screener\_cache\*\* – historic data   
8. \*\*ticker\_rotation\*\* – results from machine learning classifiers  
9. \*\*backtest\_result\*\* – summary of swing trade simulation outputs

A screenshot of a computer

AI-generated content may be incorrect.

day\_trading\_screener/

├── api/

│ ├── fmp\_client.py

│

├── db/

│ ├── \_\_init\_\_.py

│ ├── cache.py

│ ├── reader.py

│ ├── writer.py

│

├── emailer/

│ ├── notify.py

│

├── runner/

│ ├── screener\_runner.py

│

├── utils/

│ ├── exporter.py

│ ├── filters.py

│ ├── logger.py

│ ├── settings.py

│

├── output/

│ ├── screener\_results/

│ └── backtest\_results/

│

├── logs/

│ ├── scheduler.log

│ └── app.log

│

├── run\_full\_scan.py

├── watchlist\_scan.py

├── scheduler.py

├── requirements.txt

├── README.md

# Optionally:

├── start\_scheduler.bat

├── tests/

│ ├── test\_filters.py

│ ├── test\_exporter.py

Etc.