**Swing Trading ETF System – Project Outline**

**🧱 1. Core Components**

Each component handles a specific stage of the ETF rotation logic:

**a. Data Ingestion**

* Input: Historical price CSV (daily close) 🡪 scripts/fetch\_etf\_data.py (pulls unadjusted daily Close prices for all 6 EFTs using yfianance’s .history())
  + Outputs etf\_prices\_2023-2025-.csv
  + to get data from yfinance to determine buy/sell ETFs and save to Excel and csv
    - do we need both files, or can we just get into csv? It would be slightly more efficient, right and fewer files to keep. NO, Excel archived/code removed.
    - Also is there other data we can fetch from yfinance that we might use later?
      * Price divergence from 10-day high or from 30-day high)?
      * RRP trend 🡪 If RRP is rising into month-end and USFR *has already peaked*, we get confirmation (i.e., gradual rise leading toward peak?) fetch\_rrp\_data.py with pseudo code created for later as placeholder
      * From:
        + Daily usage of the Fed’s Overnight Reverse Repo Facility (ON RRP), especially:

**Falling RRP volume** → Often corresponds to declining short-term yields → Treasury ETFs like USFR may **peak soon**

**Rising or high RRP** → Implies tight liquidity and persistent yield pressure → Could keep ETF prices **suppressed**

**Monitored by** New York Fed's RRP data (updated each weekday at ~3:30 PM ET)

access data from the New York Fed's Reverse Repurchase Agreement (RRP) operations using Python, primarily through the Federal Reserve Economic Data (FRED) API.

* + - * Other possible indicates that may be more difficult to track but might provide hints (that you previously mentioned):
        + **Floating-Rate Reset Timing** (i.e., USFR resets its coupon weekly based on 90-day T-bill rates), so **price moves reflect anticipation of those resets**
        + Traders position ahead of coupon changes before **peaks**
        + Volume/discount trends

does it change (increase/decrease just before a peak?)

**Premium to NAV shrinking** or turning negative may indicate **demand waning**, and a **price peak** is near

* + - * + **Volatility** (may not indicate much since it is low)
* Preprocessing:
  + Convert Date column to datetime
  + Set as index
  + Fill missing data if needed

**b. Signal Logic**

Split into **two modules per ETF**:

* **Peak Signal Logic**
  + USFR: High in days 18–25 of month
  + Others: 1st trading day of each month
* **Low Signal Logic**
  + USFR: Lowest value in 10 days before peak
  + Others: Last trading day of **prior** month

**c. CSV Output**

* Peak signals → signals/[etf]\_post\_peak\_highs.csv
* Low signals → signals/[etf]\_post\_peak\_lows.csv

**🧠 2. Signal Generation Logic**

**USFR:**

* **Peak:** Day 18–25 of month → highest value
* **Low:** Any of prior 10 days before peak → lowest close
* ✅ Handles variation in peak timing

**Others (SGOV, BIL, TFLO, SHV, ICSH):**

* **Peak:** 1st trading day of new month (or next open trading day)
* **Low:** Last trading day of **prior** month
* ✅ Simple and repeatable pattern

**🖥️ 3. Main Scripts**

* generate\_peak\_csvs.py  
  → Generates \*\_post\_peak\_highs.csv files for all 6 ETFs
* generate\_low\_csvs.py  
  → Generates \*\_post\_peak\_lows.csv files for all 6 ETFs
* Future: main.py integrates both and calls GUI

**💡 4. GUI Features (in progress)**

* Buttons: Check Peaks, Check Lows, Refresh Data, Run All
* Display: DataFrames of recent signal dates
* Optional: Summary metrics (% gain potential, days between, alerts)

**🧰 5. Configuration Options**

* ETF list
* Rebound % threshold (e.g. 0.2%)
* Lookback windows (e.g. 10 days for USFR lows)
* Date boundaries (for peak window)

**📦 6. Folder Structure**

css

CopyEdit

/project\_root

│

├── data/

│ └── etf\_prices\_2023\_2025.csv

│

├── signals/

│ └── usfr\_post\_peak\_highs.csv

│ └── sgov\_post\_peak\_highs.csv

│ └── ...

│

├── scripts/

│ └── generate\_peak\_csvs.py

│ └── generate\_low\_csvs.py ← to be built

│ └── main.py

│

├── gui/

│ └── dashboard.py ← to be built

**🔄 7. Enhancement Ideas (future scope)**

* 📊 Add charts showing ETF price vs peak/low signals
* 🧠 Add anomaly detection (double peaks, delayed drops)
* 📈 Backtest peak-to-low strategy returns
* ⏱️ Automate monthly run with scheduler (cron or Task Scheduler)
* 📧 Email or popup alerts on new signals

Would you like me to:

* Generate a template for generate\_low\_csvs.py next?
* Update the logic in generate\_peak\_csvs.py now to follow your clarified low/peak rules for each ETF?
* Start planning main.py to glue both peak and low logic together?

Let me know how you'd like to proceed.