# End of Day Status Report

\*This document helps track progress and inform the next day's activities. If work is handed off to a new Claude AI session, this helps the new instance understand the previous state.\*

**TEMPLATE**:

## 1. Current Status

- Key achievements and current state of the project

- Status of major components

- Critical metrics (e.g., test coverage, successful API calls)

## 2. Recent Improvements

- Completed tasks since last report

- Bug fixes

- Code improvements

- New features added

## 3. Current Issues

- Active bugs or problems

- Performance issues

- Integration challenges

- Resource constraints

## 4. Immediate To-Do List (Next 1-2 Sessions)

- Prioritized, actionable tasks

- Bug fixes needed

- Tests to be written

- Expected complexity (High/Medium/Low)

- Dependencies or blockers

## 5. Planned Improvements (Longer Term)

- Future features

- Architecture changes

- Performance optimizations

- Technical debt to address

## 6. Infrastructure & Tooling Needs

- CI/CD requirements

- Testing infrastructure

- Monitoring needs

- Development environment improvements

## 7. Open Questions

- Design decisions needed

- Technical choices to be made

- Resource allocation questions

- Risk assessment needs

## 8. Notes for Next Session

- Context needed for next work session

- Suggested starting points

- Important caveats or considerations

***Latest report is below.***

# **End of Day Status Report - March 4, 2025**

## **1. Current Status**

* Successfully implemented Yahoo Finance as a data source with proper fallback mechanisms
* Created MarketDataCollector class to coordinate between multiple sources
* Implemented batching and throttling to handle API rate limits
* Successfully tested the new system, confirming it can collect stock data despite rate limits
* Current approach saves partial results as CSV files after each batch
* Progress on collecting NASDAQ-100 data is now more reliable but slower due to rate limit handling

## **2. Recent Improvements**

* Added YahooFinanceDataSource implementation
* Created MarketDataCollector to manage multiple data sources with fallback logic
* Implemented batched processing with 5 tickers per batch
* Added progressive delays (5s between tickers, 30s between batches)
* Added exponential backoff for source fallbacks
* Implemented partial results saving after each batch
* Improved error handling and logging

## **3. Current Issues**

* All three data sources (Yahoo Finance, FMP, Finnhub) have rate limiting issues
* Yahoo Finance is getting "429 Too Many Requests" errors consistently
* FMP API also reaching rate limits quickly
* Full data collection is slow due to necessary throttling and delays
* No caching mechanism to avoid repeated requests for the same stock

## **4. Immediate To-Do List (Next 1-2 Sessions)**

* Implement caching mechanism to store successful responses (Complexity: Medium)
* Create AlphaVantage data source as an additional option (Complexity: Low)
* Develop focused ticker list with only top 20-30 stocks by market cap (Complexity: Low)
* Begin script generation implementation (Complexity: High)
* Create unit tests for the Yahoo Finance implementation (Complexity: Medium)

## **5. Planned Improvements (Longer Term)**

* Implement a database for storing historical stock data
* Create a more sophisticated scheduling system
* Implement more detailed technical analysis
* Add market sentiment analysis from news sources
* Develop visualization components for the script
* Consider paid API alternatives if free options continue to have limitations

## **6. Infrastructure & Tooling Needs**

* Consider setting up a Redis or SQLite cache for API responses
* Add monitoring for API rate limits and failures
* Set up a cron job for running daily collection after market close
* Improve Docker configuration for production deployment
* Implement automated cleanup of temporary/partial files

## **7. Open Questions**

* Should we focus only on the top market cap companies to reduce API calls?
* Is it worth investing in a paid API service for more reliable data?
* How much historical data should we maintain for each stock?
* What's our strategy for weekends and market holidays?
* Should we add functionality to backfill missing days?

## **8. Notes for Next Session**

* The current implementation successfully collects data but is slow due to rate limiting
* Prioritize implementing the ScriptGenerator class using the collected stock data
* Use the partial\_top\_movers and partial\_bottom\_movers CSV files for script generation
* Consider adjusting the throttling parameters if collection is too slow
* Remember that Yahoo Finance gets data successfully for NASDAQ-100 tickers but hits rate limits quickly