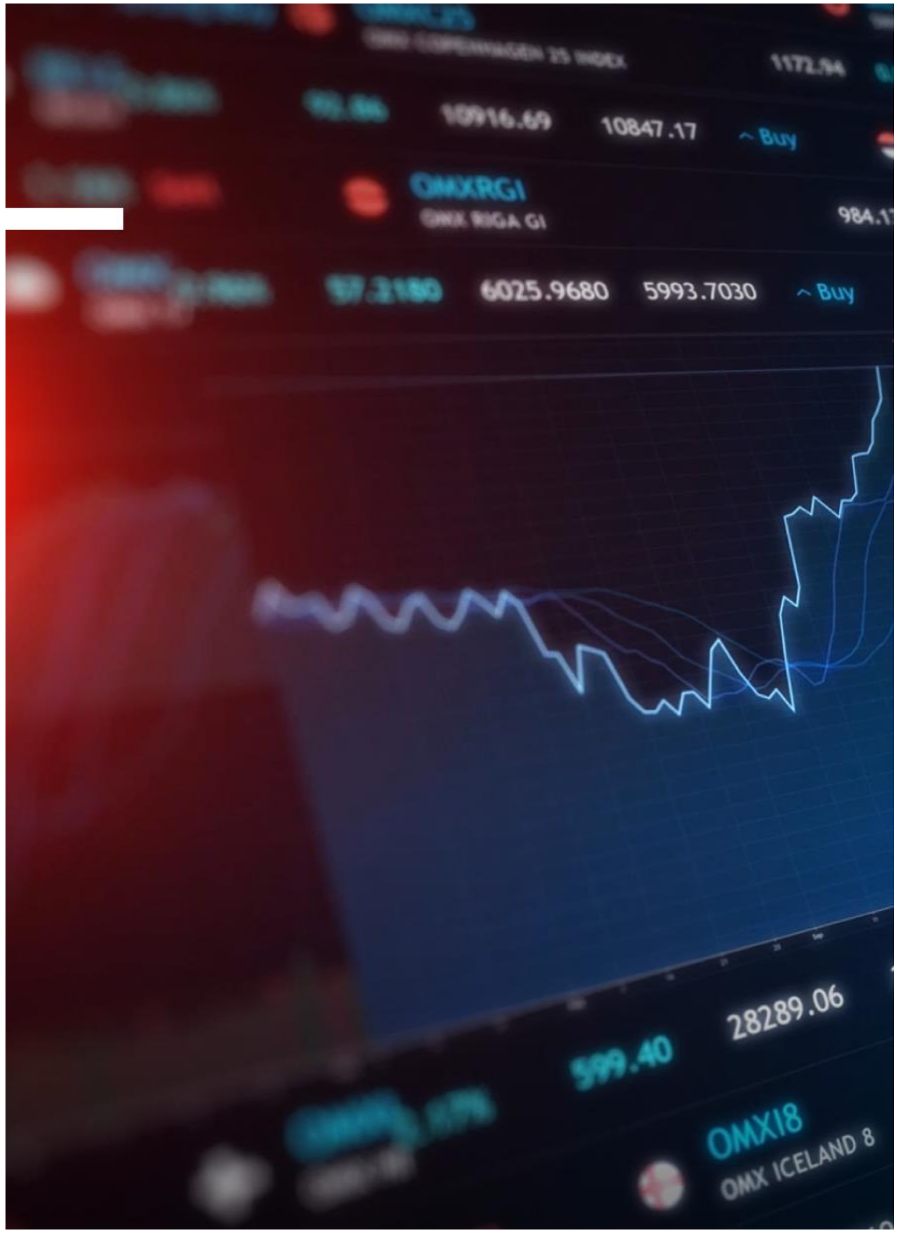


REAL-TIME STOCK TRADING SIMULATOR

EECE 2560

Noah Assam, Chance Bowman

Introduction

Project Scope

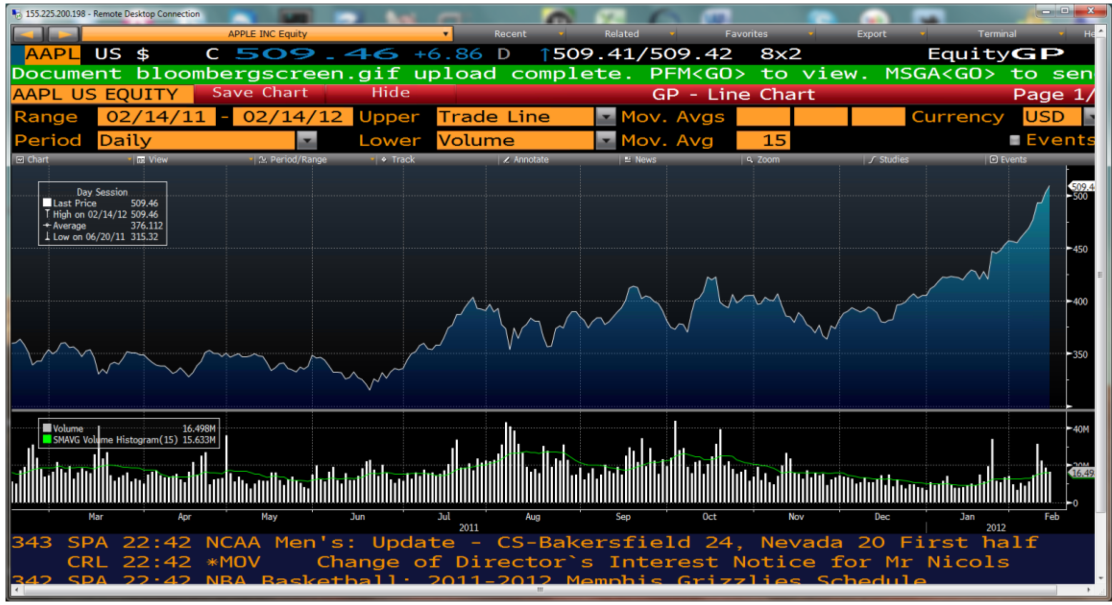
- Develop a stock trading simulator that enables users to buy and sell stocks based on predicted trends derived from data.
- Address the challenge of making trading decisions in a volatile stock market.
- A functional trading simulator with real time data integration.

Objectives and Goals

- To develop a user interface for real-time stock trading.
- To implement, and refine, greedy algorithms that optimize trading decisions.
- To use historical stock data to verify algorithmic accuracy and refine strategies.
- The outcomes include a functioning system, proper documentation, and a final project report.

Literature Review

- Tools like Bloomberg Terminal provide real time market data for trading decisions.
- StockTrak and Wall Street Survivor are platforms that provide a virtual trading experience with real time market data.
- Drawing from techniques such as moving average crossovers, mean reversion, and machine learning based strategies which are widely used for automating trading decisions.
- Existing platforms can be complex for new users. Project aims to deliver a simple user interface with intuitive design for strategy testing and execution.
- Drawing inspiration from StockTrak, the project aims to combine real time market feeds with historical data for a comprehensive trading simulation.



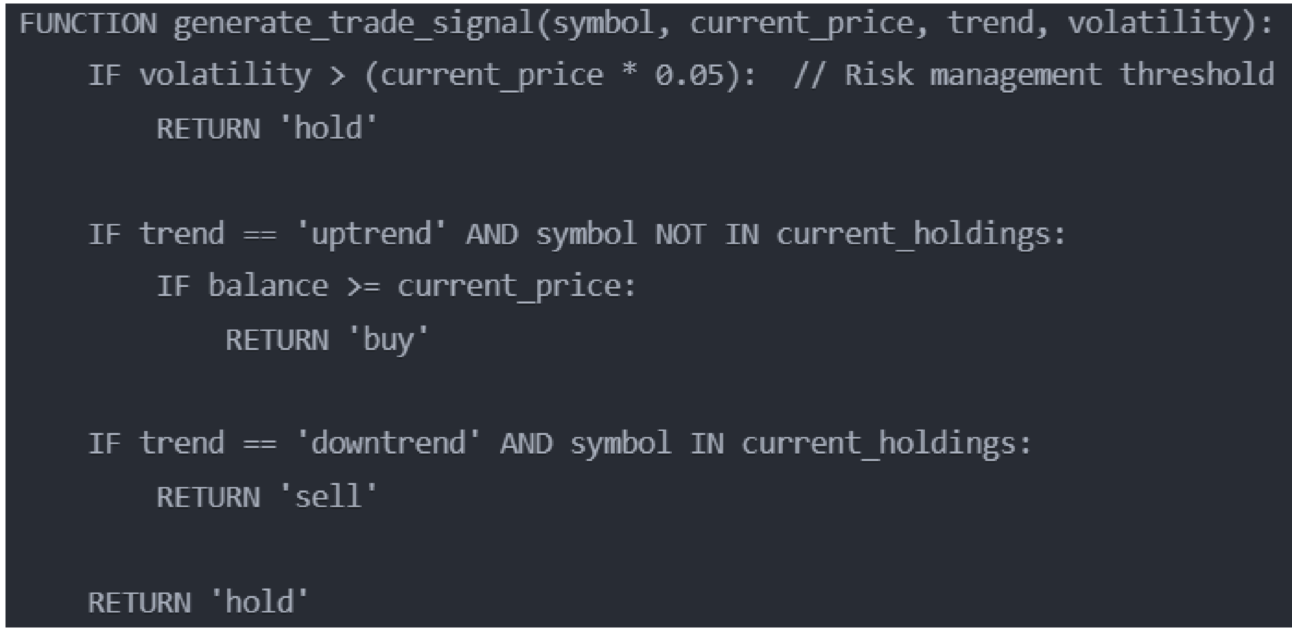
```
AAPL US $      C 509.46 +6.86 D  ↑509.41/509.42  8x2          EquityGP
Document bloombergscreen.gif upload complete. PFM<GO> to view. MSGA<GO> to sen
AAPL US EQUITY   Save Chart   Hide              GP - Line Chart          Page 1/
Range   02/14/11 - 02/14/12  Upper  Trade Line  Mov. Avgs              Currency USD
Period  Daily                Lower  Volume      Mov. Avg   15          Events
343 SPA 22:42 NCAA Men's: Update - CS-Bakersfield 24, Nevada 20 First half
    CRL 22:42 *MOV     Change of Director`s Interest Notice for Mr Nicols
342 SPA 22:42 NBA Basketball: 2011-2012 Memphis Grizzlies Schedule
```

Project Timeline

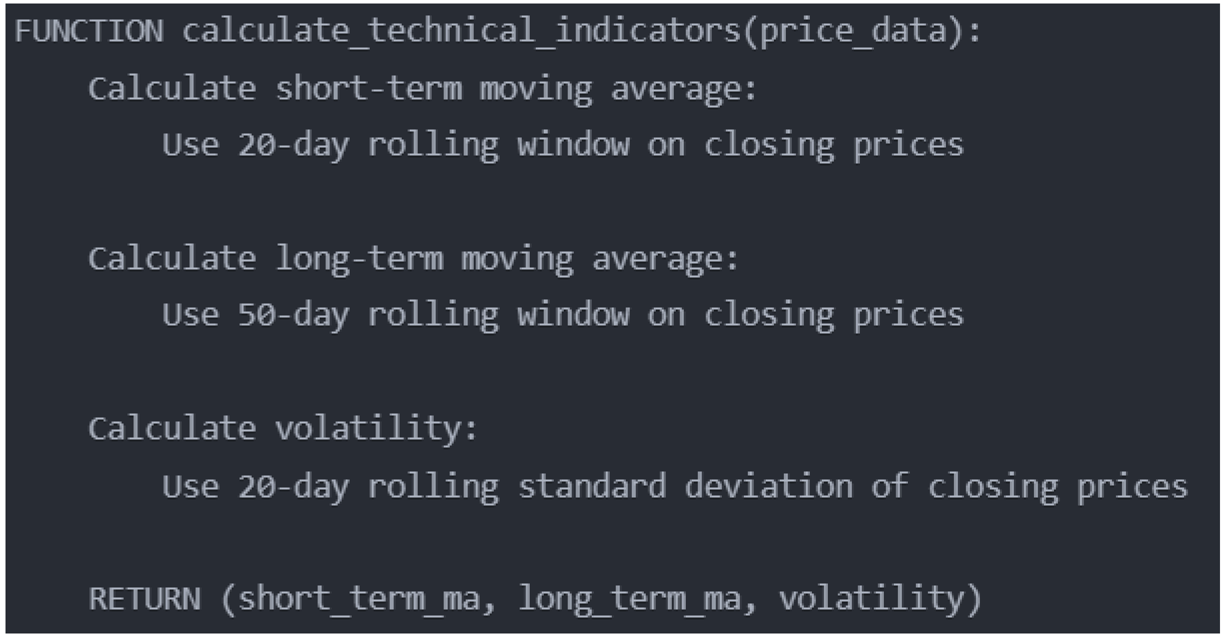
- Week 1 defined project scope, establish team roles, and outline skills/tools.
- Week 2 Research and designing system architecture.
- Week 3 Begins UI development and greedy algorithm prototyping, and some initial tests with historical data.
- Week 4 Further refines greedy algorithm, implement a more advanced trade logic, and begin backend integration.
- Week 5 Develop graph algorithms, if necessary, for trend analysis. Complete database integration.
- Week 6 Testing and debugging.

Methodology

- For data collection utilized the python library yfinance to fetch historical stock data
- The technical analysis calculated short-term (20-day), long-term (50-day) moving averages and volatility.
- The trend analysis classified market as uptrend, downtrend or neutral using moving averages.
- Generate signals (buy, sell, hold) based on trends and volatility thresholds.
- Risk management is to avoid trades with >5% volatility.
- Employ greedy algorithms to optimize trades.



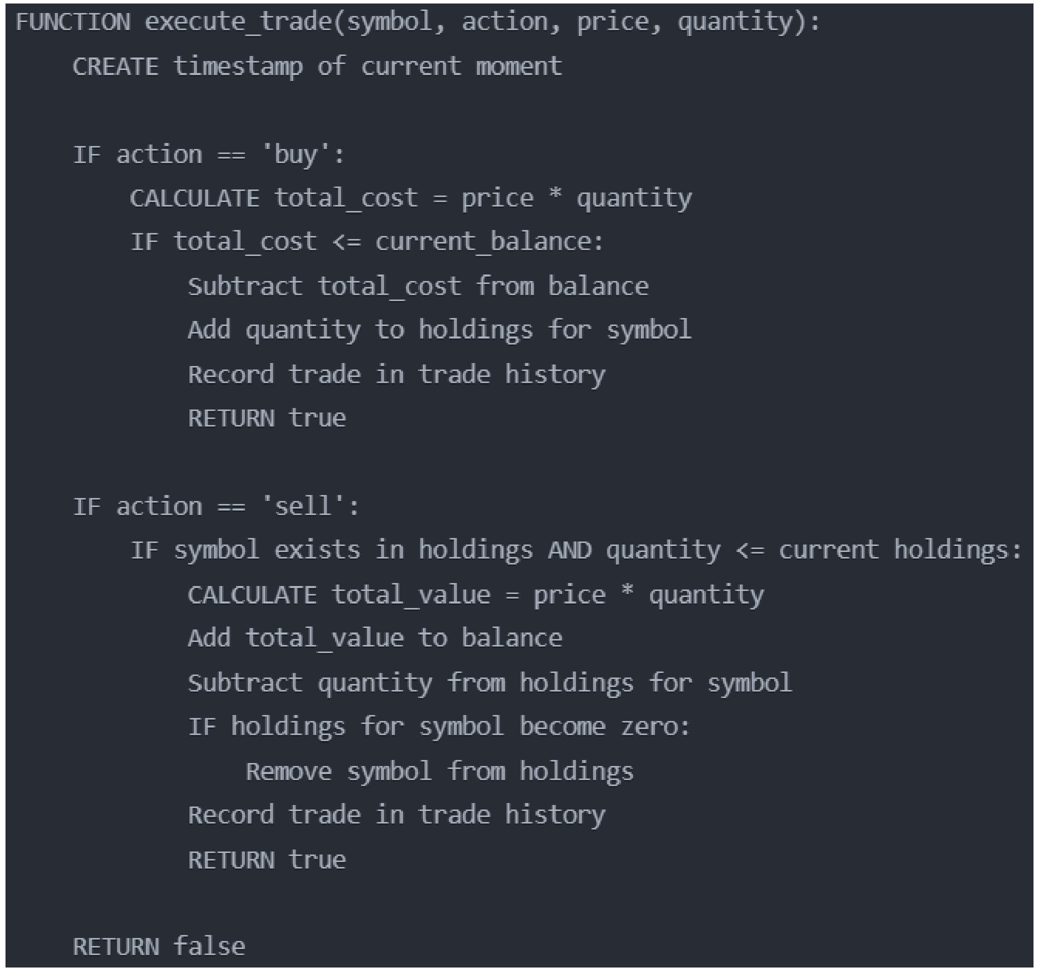
```
FUNCTION generate_trade_signal(symbol, current_price, trend, volatility):
    IF volatility > (current_price * 0.05):  // Risk management threshold
        RETURN 'hold'

    IF trend == 'uptrend' AND symbol NOT IN current_holdings:
        IF balance >= current_price:
            RETURN 'buy'

    IF trend == 'downtrend' AND symbol IN current_holdings:
        RETURN 'sell'

    RETURN 'hold'
```



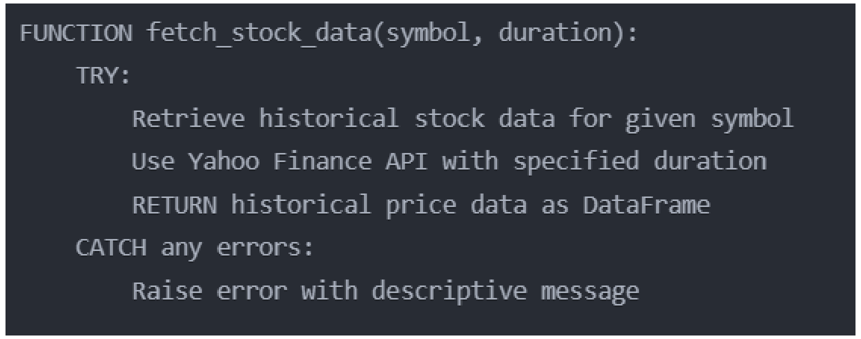
```
FUNCTION calculate_technical_indicators(price_data):
    Calculate short-term moving average:
        Use 20-day rolling window on closing prices

    Calculate long-term moving average:
        Use 50-day rolling window on closing prices

    Calculate volatility:
        Use 20-day rolling standard deviation of closing prices

    RETURN (short_term_ma, long_term_ma, volatility)
```

Methodology

- Utilized python library pandas DataFrame to store historical stock prices for computation.
- Utilized dictionaries as holdings to track quantities of owned stocks.
- Utilized lists to log trade history for reporting.
- Utilized numpy arrays for efficient numerical computations for moving averages and volatility.
- Overall Complexity: O(n).



```
FUNCTION execute_trade(symbol, action, price, quantity):
    CREATE timestamp of current moment

    IF action == 'buy':
        CALCULATE total_cost = price * quantity
        IF total_cost <= current_balance:
            Subtract total_cost from balance
            Add quantity to holdings for symbol
            Record trade in trade history
            RETURN true

    IF action == 'sell':
        IF symbol exists in holdings AND quantity <= current holdings:
            CALCULATE total_value = price * quantity
            Add total_value to balance
            Subtract quantity from holdings for symbol
            IF holdings for symbol become zero:
                Remove symbol from holdings
            Record trade in trade history
            RETURN true

    RETURN false
```

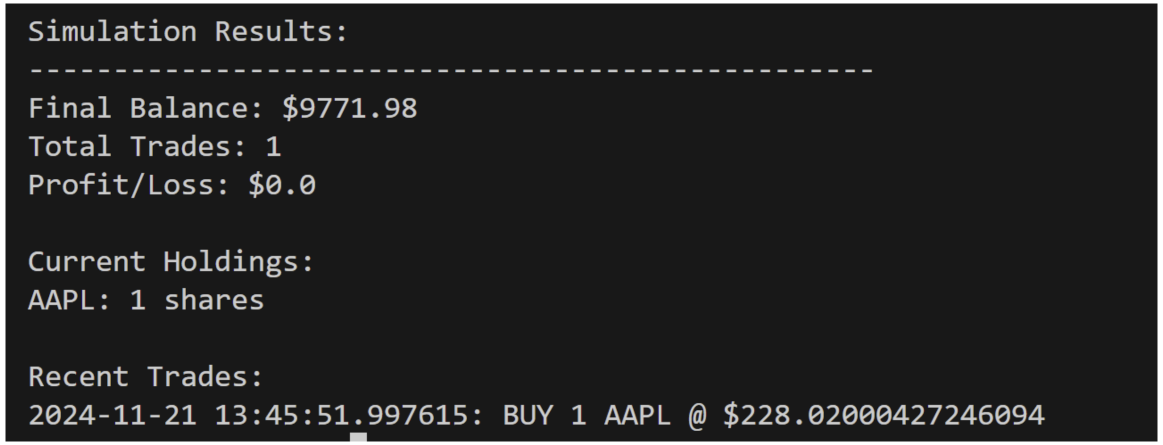


```
FUNCTION fetch_stock_data(symbol, duration):
    TRY:
        Retrieve historical stock data for given symbol
        Use Yahoo Finance API with specified duration
        RETURN historical price data as DataFrame
    CATCH any errors:
        Raise error with descriptive message
```

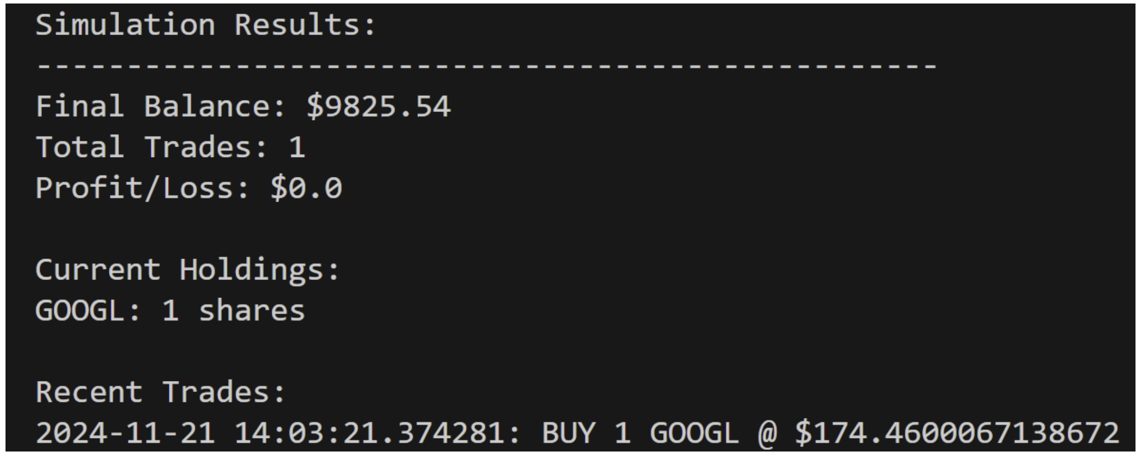
Time Spent on Project

3 Hours/Week X 7 Weeks = 21 Hours

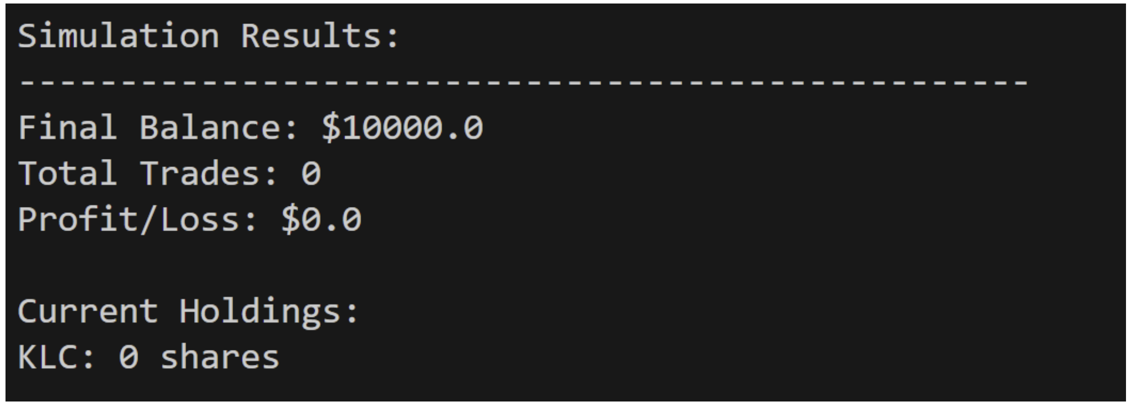
Analysis and Results



```
Simulation Results:
--------------------------------------------------
Final Balance: $9771.98
Total Trades: 1
Profit/Loss: $0.0

Current Holdings:
AAPL: 1 shares

Recent Trades:
2024-11-21 13:45:51.997615: BUY 1 AAPL @ $228.02000427246094
```



```
Simulation Results:
--------------------------------------------------
Final Balance: $9825.54
Total Trades: 1
Profit/Loss: $0.0

Current Holdings:
GOOGL: 1 shares

Recent Trades:
2024-11-21 14:03:21.374281: BUY 1 GOOGL @ $174.4600067138672
```



```
Simulation Results:
--------------------------------------------------
Final Balance: $10000.0
Total Trades: 0
Profit/Loss: $0.0

Current Holdings:
KLC: 0 shares
```

Discussion

Data-driven trading strategies can simplify decision-making in the stock market, making trading more accessible.

This simulator may also serve as an educational platform for understanding trading algorithms and market trends.

The simulator’s backend has not yet fully been implemented, restricting data storage between encounters. The lack of detailed performance metrics may also limit a complete assessment of the simulator’s efficacy.

The greedy algorithm’s simplicity overlooks more complex market conditions, where dynamic programming or another alternative may prove more efficient.

Conclusion

When paired with basic technical indicators (i.e., moving averages), greedy algorithms can facilitate effective trading decisions.

The development of the interface demonstrates how trading tools can be made more accessible, bridging the gap between complex financial systems and general consumers.

As for future research:

- Using more enhanced algorithms, as a way to implement more sophisticated strategies (i.e., reinforcement learning for the adaption to varying market conditions.
- Using more enhanced risk management, such as RSI, ADX, or Bollinger Bands.
- Creating a better interface.

References

https://www.bloomberg.com/professional/products/bloomberg-terminal/#terminal-in-action

Basics of Algorithmic Trading: Concepts and Examples

Ultimate Guide to Algorithmic Trading Strategies - TradingCanyon