Backtesting Project Specification Document

---

1. Project Overview

This document outlines the requirements and expectations for a backtesting project. The goal of the project is to simulate how a specific stock trading strategy would have performed in the past, using historical stock data.

Definition of Terms

- **Backtesting**: The process of testing a trading strategy by applying it to historical market data to see how it would have performed.
- **Rule**: A defined condition or set of conditions that triggers a trade (buy or sell). For example, "buy when a stock increases three days in a row."
- **Buy Signal**: The condition under which the system should buy a stock.
- **Sell Signal**: The condition under which the system should sell a stock.
- **Stock Universe**: The set of stocks that the strategy will be tested on.

Standard Candlesticks

Normal stock price reporting—also known as standard candlestick data—shows the actual market prices at specific intervals (e.g., daily). Each data point includes:

- **Open: The price at the beginning of the trading period**
- **High: The highest price reached during the period**
- **Low: The lowest price reached during the period**
- **Close: The price at the end of the trading period**
- **Volume (optional): The number of shares traded**

This format reflects true historical prices and is what you typically get from sources like Yahoo Finance.

Heiken Ashi Candlesticks

Heiken Ashi (Japanese for “average bar”) is a type of candlestick charting that smooths out price movements to better show trends. Unlike standard candlesticks, Heiken Ashi candles are calculated using formulas that incorporate previous periods’ prices:

- **HA Close = (Open + High + Low + Close) / 4**
- **HA Open = (Previous HA Open + Previous HA Close) / 2**
- **HA High = max(High, HA Open, HA Close)**
- **HA Low = min(Low, HA Open, HA Close)**

Heiken Ashi does not represent actual traded prices. It's mainly used to reduce noise in the data, which can be helpful for building our strategy.

The goal of this project is not just to create a profitable system, but to learn the process of designing and testing an algorithm.

Technical Terms

A **moving average** is a technical indicator used to smooth out price data by averaging a stock's price over a certain number of past days.

- **Simple Moving Average (SMA)**: This is the most basic type. It takes the closing prices of a stock over a specific number of days and calculates their average. There are more complex moving averages, but this is all we will need for this strategy

MACD (Moving Average Convergence Divergence)

The **MACD** is a momentum indicator that shows the **relationship between two moving averages** of a stock's price.

It consists of three components:

- **MACD Line** = For example: 12-day SMA − 26-day SMA
- **Signal Line** = 9-day SMA of the MACD Line
- **Histogram** = MACD Line − Signal Line

Lines and histograms can be translated to a table with each date as a row and Histogram amount as a number

---

2. Strategy Description

Heiken Ashi Trading Rule

For this Strategy, we'll be using Heiken Ashi just to determine when to buy and sell. However, when we are actually executing the action of buying or selling, we must buy and sell the 'Open' price that can be found in the yfinance data, which is standard format. You need to replicate and transform the standard candlestick into HA to determine the buy/sell dates.

- **Holding Period Constraints**: No minimum or maximum hold period; rely solely on buy/sell conditions.
    - **Sell the entire position at the end of your simulation so we can evaluate our cumulative return**
- **Buy Rules** (buy when **all** these conditions hold true)
    - It is on Monday, Wednesday or Friday
    - We currently have <10,000 in the stock already (Our case limit)
    - This day is not a day to sell our position
- **Sell Rules** (sell **all our position** when **most** of these conditions are true)
    - The close price is lower than the HA 20-week (100-day) Moving Average
    - The HA 10-week (50-day) Moving Average has not increased from yesterday
    - The Moving Average Convergence Divergence (MACD) is negative or decreasing

Implement the first two sell rules and evaluate and then save that code, then work on the MACD rule.

---

3. Stock Universe

The backtest should be run on the following 24 stocks:

- TSM, QCOM, SPGI, TJX, ADI, WM, MCK, SHW, RSG, TEAM, FICO, RMD,
- MPWR, TSCO, TYL, EME, JBHT, MANH, SAIA, FSS, UFPT, WS, WTTR, VMD

You do not need to test every stock in the list. Choose at least **three to five** stocks for initial testing to validate your algorithm.

---

4. Data Requirements

- **Source**: Use the yfinance Python library to pull historical data.

- **Frequency**: Daily closing prices.
- **Fields Required**: At a minimum, use the closing price; optionally, use high, low, open, and volume if desired.
- **Time Range**: At least 5 years of daily historical data (start with 3 years)
- **Handling Missing Data**: Stocks with insufficient data should be skipped (for ease)

---

5. Architecture and Implementation Guidelines

- **Language**: Python
- **Suggested Libraries**: pandas, numpy, matplotlib, yfinance

Recommended File Structure

- data_loader.py: Downloads and prepares historical data
- strategy.py: Implements buy/sell logic
- backtest.py: Executes trades based on signals

---

6. Performance Metrics

Calculate and report the following:

- Total Return
- Number of Trades
- Win Rate (percentage of profitable trades)
- Average Holding Period
- Maximum Drawdown (largest peak-to-trough decline)

---

7. Output Requirements

- **Summary Report**: Include key performance metrics and trade logs
- **Visualizations**:
    - Cumulative returns over time

    - Trade entry and exit markers on price charts
    - Drawdown chart
    - An image of the stock price, with green arrows pointing to the line on days where we indicated to buy and red arrows pointing to the line indicating sell days
- **Logs**:
    - List of trades with entry/exit dates, prices, and profit/loss

---

8. Evaluation Criteria

- Correctness of implementation
- Clarity and organization of code
- Accuracy and completeness of performance metrics
- Insightful analysis of results (did the rule perform well? why or why not?)

---

9. Required Software

Install the following tools and libraries before starting the project:

- VSCode
- Python 3.8 or higher
- pip (Python package manager)
- Jupyter Notebook Extension (optional)
- Python libraries:
    - pandas
    - numpy
    - matplotlib
    - yfinance
- Power BI (for phase 2: visualization and reporting)

Use pip install to install the required Python libraries, for example:

pip install pandas numpy matplotlib yfinance

---

10. Collaboration, AI, & GitHub Resources

To work effectively as a team of two developers, follow these best practices:

- **Divide the Work**: Assign specific components (e.g., data loading, strategy logic, visualization) to each person.
- **Communicate Frequently**: Discuss issues, updates, and decisions at least once per day.
- **Use GitHub for Version Control (option)**:
    - Create a shared GitHub repository
    - Use branches for separate features
    - Commit and push your changes frequently
    - Review and merge code using pull requests
- **Use AI tools to help you explain concepts, generate code, debug errors, etc.**
