

Group 16: Indicator Builder

Final Report

by

Simon Huang s355huan

Charlie Cao y293cao

Jack Liu zj5liu

April 2025

Table of Contents

Table of Contents	2
1. Introduction	3
1.1 Product Overview	3
1.2 Conventions	4
1.3 Glossary of Terms	4
2. Domain Model	5
2.1 Domain Model with Superimposed World Diagram	5
3. Use Case Diagram	6
3.1 Viewing a Strategy Inside the UI	6
3.2 Configuring the Strategy's Properties	7
4. Assumptions, Exceptions, Variations	9
4.1 Assumptions	9
4.2 Exceptions	9
4.3 Variations	10
5. User Manual (Use Cases)	11
5.1 Use Case 1	11
5.2 Use Case 2	13
5.3 Use Case 3	14
5.4 Use Case 4	15
5.5 Use Case 5	16
6. Final Software Design	18
6.1 Frontend	18
6.2 Backend	18
6.3 Deployment	19
7. Verification and Validation	20

1. Introduction

1.1 Product Overview

Indicator Builder is a trading tool that allows both amateur and experienced traders to experiment with indicators and trading strategies. Designed to simplify complex market analysis, the indicators provide real-time suggested **buy** and **sell** signals derived from trading strategies.

By automating the detection of patterns and trends in financial data, **Indicator Builder** allows traders to make informed decisions in volatile markets where human reaction times could fall short. The tool integrates the data displayed by indicators and asset price data, overlaying insights directly onto price charts.

1.2 Conventions

The following conventions are used:

- Times New Roman is used for normal text.
- **Bold** is used for section names.
- Numbered lists that are not part of a heading represent a sequence of steps to be taken to achieve a goal.
- Bullet lists are an assortment of unordered, related ideas.

1.3 Glossary of Terms

The following terms are used in the manual:

- *tool* – refers to the system or program, Indicator Builder.
- *user* – refers to the person using the tool.
- *instrument* – refers to any tradable financial asset, such as a stock or bond.
- *indicator* – mathematical calculations based on financial data that can provide analytics that help the user make better decisions about their trades.
- *strategy* – automated system of trades based on the values of one or more indicators
- *parameter* – inputs for indicators or strategies to customize for specific needs

2. Domain Model

2.1 Domain Model with Superimposed World Diagram

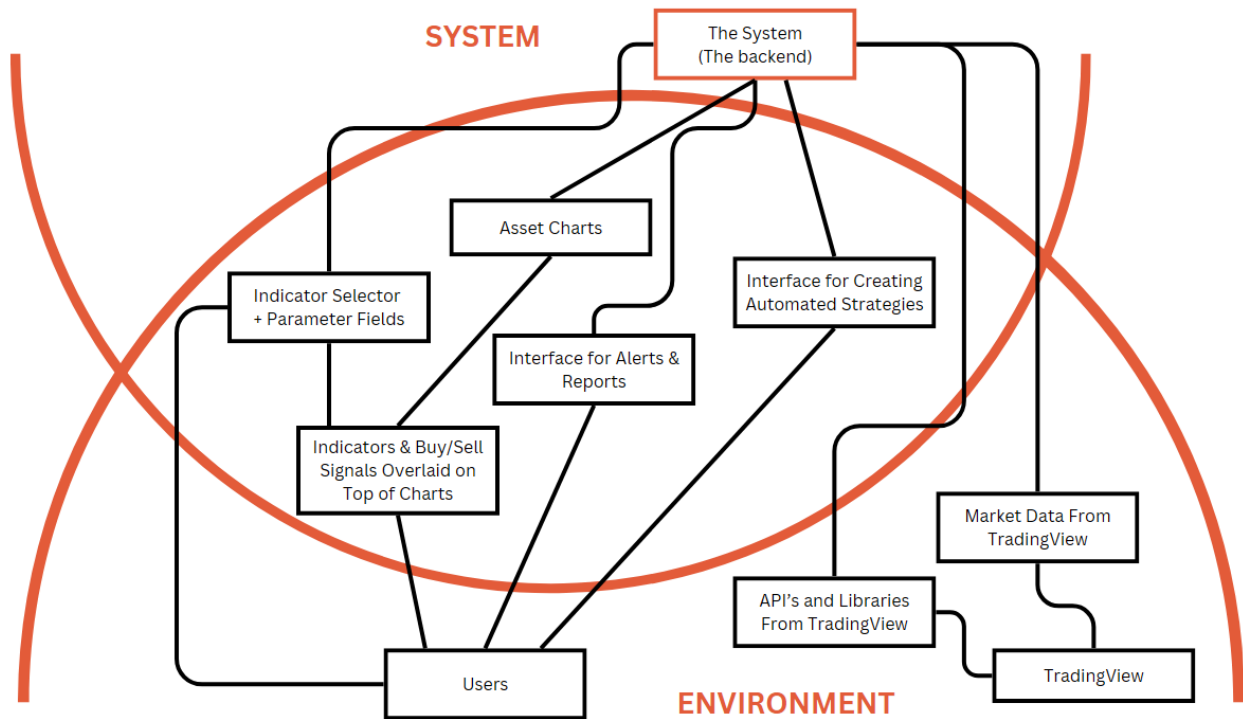


Figure 2.1: Domain Model

3. Use Case Diagram

3.1 Viewing a Strategy Inside the UI

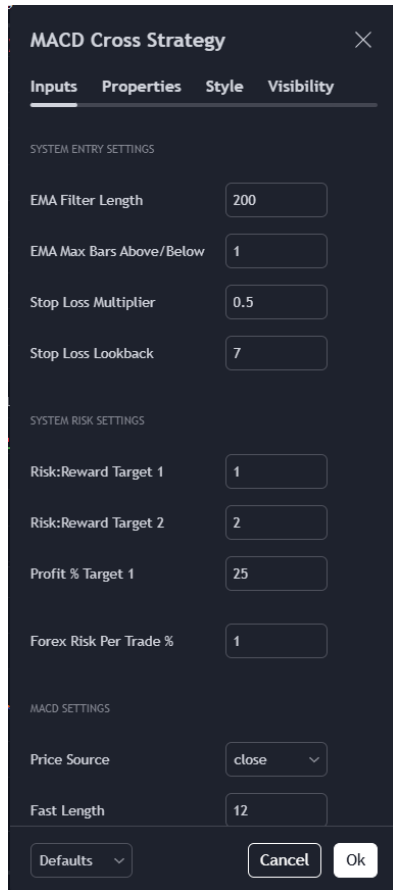


Figure 3.1: Sample MACD Strategy

The Strategy UI displays the following items

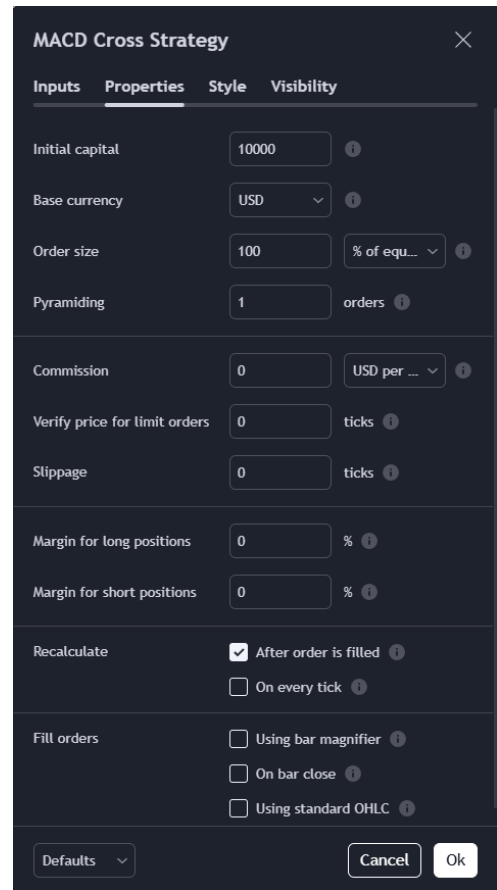
- candles - prices depicting the most recent change in price of the instrument
- indicator - the graph below the main candles representing the values of one or more indicators that are used in the strategy
- longs - positions that the strategy suggests longing (buying) the instrument
- shorts - positions that the strategy suggests selling the instrument
- long exits - positions to stop the most recent long
- short exits - positions to stop the most recent short

3.2 Configuring the Strategy's Properties



The image shows the 'Inputs' tab of the 'MACD Cross Strategy' configuration window. It features four sections: 'SYSTEM ENTRY SETTINGS' with fields for EMA Filter Length (200), EMA Max Bars Above/Below (1), Stop Loss Multiplier (0.5), and Stop Loss Lookback (7); 'SYSTEM RISK SETTINGS' with fields for Risk:Reward Target 1 (1), Risk:Reward Target 2 (2), Profit % Target 1 (25), and Forex Risk Per Trade % (1); 'MACD SETTINGS' with fields for Price Source (close) and Fast Length (12); and a bottom section with a 'Defaults' dropdown, 'Cancel', and 'Ok' buttons.

Figure 3.2: Strategy Inputs



The image shows the 'Properties' tab of the 'MACD Cross Strategy' configuration window. It features several sections: 'Initial capital' (10000), 'Base currency' (USD), 'Order size' (100) with a '% of equ...' dropdown, 'Pyramiding' (1) with an 'orders' dropdown, 'Commission' (0) with a 'USD per ...' dropdown, 'Verify price for limit orders' (0) with a 'ticks' dropdown, 'Slippage' (0) with a 'ticks' dropdown, 'Margin for long positions' (0) with a '%' dropdown, 'Margin for short positions' (0) with a '%' dropdown, 'Recalculate' with a checked 'After order is filled' checkbox and an unchecked 'On every tick' checkbox, and 'Fill orders' with three unchecked checkboxes: 'Using bar magnifier', 'On bar close', and 'Using standard OHLC'. The bottom section includes a 'Defaults' dropdown, 'Cancel', and 'Ok' buttons.

Figure 3.3: Strategy Properties

Clicking on the strategy will display configurations, which include

- Inputs
 - EMA Filter Length - number of days to track the exponential moving average
 - EMA Max Bars Above/Below - number of bars above or below the EMA to initiate trading

- Stop Loss Multiplier - the maximum percent to lose before stop loss is triggered
- Stop Loss Lookback - the number of bars the stop loss multiplier refers to
- Risk-Reward Target 1 - risk-reward ratio as a first target
- Risk-Reward Target 2 - risk-reward ratio as a second target
- Profit % Target 1 - amount to take profit after target 1 is hit
- Forex Risk Per Trade - percentage of available capital to trade
- Properties
 - Initial capital - amount to start trading with
 - Base currency - currency of the capital
 - Order size - the number of shares to trade, or a percentage of equity
 - Pyramiding - maximum number of entries in a specific direction
 - Commission - cost per trade
 - Verify price for limit orders - number of ticks past limit price
 - Slippage - number of ticks added to fill price of market/stop orders

Note 1: Style and Visibility are also present in this tab, but they do not affect how the strategy is executed and are purely based on preference

Note 2: Different strategies may have different inputs and properties, the displayed inputs and properties are only for the sample MACD Crossover Strategy.

4. Assumptions, Exceptions, Variations

4.1 Assumptions

- All dependencies are operational and available for the tool to use.
- The indicator script does not timeout and finishes executing within the resource limits of the backend.
- The user is familiar with trading indicators, including what various parameters mean and the appropriate values for them.
- The user is using the indicator on an instrument that is available to trade (e.g. not S&P500 on weekends).
- All dependencies of the tool do not make changes to their platform that breaks compatibility with the tool.

4.2 Exceptions

- An unexpected trading halt occurs on a stock exchange.
- Any dependency becoming temporarily unreachable or unavailable.
- The indicator script does not finish executing within the time limits or exceeds the resource limits of the backend.
- The user sets nonsensical parameters in any of the use cases.
- The user attempts to select an instrument that is incompatible with the indicator or an instrument that is not available to trade.

- The pricing data for an instrument is incomplete or inaccurate. The tool's functionalities will still work, but may produce incorrect results.
- Not enough liquidity for an instrument to display correct information on the pricing charts. The tool's functionalities will still work, but may produce incorrect or inaccurate results.

Further details about exceptions can be found in **Section 5: Use Cases**

4.3 Variations

- User sets parameters of an indicator in a way such that it generates the same signals as another indicator.
- User captures the information on screen directly to obtain the information that would be included in a report, instead of using the generate report feature.

Further details about variations and alternatives can be found in **Section 5: Use Cases**

5. User Manual (Use Cases)

5.1 Use Case 1

View the indicator data and generated buy/sell signals for an instrument, overlaid on top of the price chart

5.1.1 Typical Usage Scenario

1. Select an indicator and instrument to be used for this scenario
2. Set the appropriate parameters for the indicator
3. Confirm settings, causing the tool to start calculations
4. User views the indicator data, and the generated signals overlaid on the price chart of the instrument

5.1.2 Exceptional Scenarios

- The indicator script does not finish executing within the time limit or other resource limits allowed by the backend, whether due to the parameters set by the user or otherwise. The tool will display an error then ask the user to retry.
- The user attempts to enter invalid values as parameters for indicator calculations. The tool will reject invalid inputs that will cause errors in calculations. However, inputs that are nonsensical but not invalid will be allowed, and will result in the tool generating nonsensical output. It is up to the user to avoid such cases.

- The user attempts to select an instrument that is incompatible with the indicator or an instrument that is not available to trade. The tool will display an error then ask the user to retry with a different selection.

5.1.3 Alternative Scenarios

- The user sets parameters of an indicator in a way such that it generates the same signals as a different indicator.

5.2 Use Case 2

View the moving average (MA) for an instrument and the relative strength index (RSI) to compare generated signals

5.2.1 Typical Usage Scenario

1. Select an instrument to be used for this scenario
2. Set the appropriate calculation parameters
3. Confirm settings, causing the tool to start calculations
4. User views the signals generated by MA and RSI overlaid on price chart of the instrument, then make observations about the signals

5.2.2 Exceptional Scenarios

- The indicator script does not finish executing within the time limit or other resource limits allowed by the backend, whether due to the parameters set by the user or otherwise. The tool will display an error then ask the user to retry.
- The user attempts to enter invalid parameters for the MA or RSI calculations. The tool will reject invalid inputs that will cause errors in calculations. However, inputs that are nonsensical but not invalid will be allowed, and will result in the tool generating nonsensical output. It is up to the user to avoid such cases.

5.3 Use Case 3

Request for an alert when an instrument reaches a certain price, moving average, or indicator threshold

5.3.1 Typical Usage Scenario

1. Select the type of event to receive an alert for
2. Set the threshold value to receive an alert, and select the indicator if the event type is an indicator threshold
3. Confirm settings, causing the tool to register the event for alerts
4. User will now receive notifications whenever the criteria for alert is met

5.3.2 Exceptional Scenarios

- The user attempts to request alerts for an indicator threshold with an instrument that is incompatible with the indicator. The tool will display an error then ask the user to retry with a different selection.

5.3.3 Alternative Scenarios

- The user sets a threshold for an event that is impossible to be met. The tool will register such an event for alerts, but no alert will ever be sent.

5.4 Use Case 4

Generate a downloadable report summarizing key technical indicators for selected instruments within a time period

5.4.1 Typical Usage Scenario

1. Select the instruments to be used for the report
2. Select the indicators to be included in the report and set the parameters for them
3. Set the start and end time for the report period
4. Confirm settings, causing the tool to perform calculations and generate report
5. User will be able to download the report once it is generated

5.4.2 Exceptional Scenarios

- The user selects incompatible instruments and indicators for the report. The tool will display an error then ask the user to retry with a different selection.
- The user enters invalid values as parameters for indicator calculations. The tool may generate a report with nonsensical information, or encounter an error while generating the report then ask the user to retry.

5.4.3 Alternative Scenarios

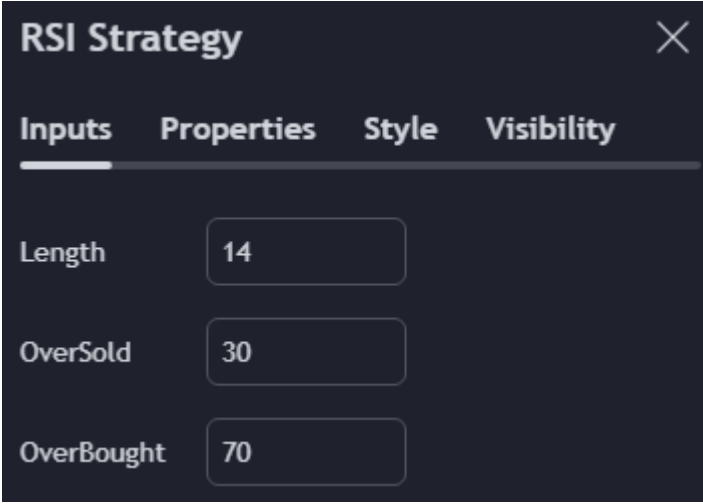
- The user may capture the information on screen directly to obtain the information that would be included in a report, instead of using the generate report feature.

5.5 Use Case 5

Create a trading strategy for an instrument based on indicators and buy/sell signals, then backtest

5.5.1 Typical Usage Scenario

1. Select the instrument, indicators and strategy to be used
2. Set the parameters for the indicators and the strategy



The screenshot shows a window titled "RSI Strategy" with a close button (X) in the top right corner. Below the title bar are four tabs: "Inputs", "Properties", "Style", and "Visibility". The "Inputs" tab is selected and highlighted with a white underline. Under the "Inputs" tab, there are three input fields: "Length" with a value of 14, "OverSold" with a value of 30, and "OverBought" with a value of 70.

Fig. 5.1: Example of strategy parameter input

3. Confirm settings, causing the tool to create the strategy based on the inputs
4. User can then backtest the strategy or export it to be used in trading

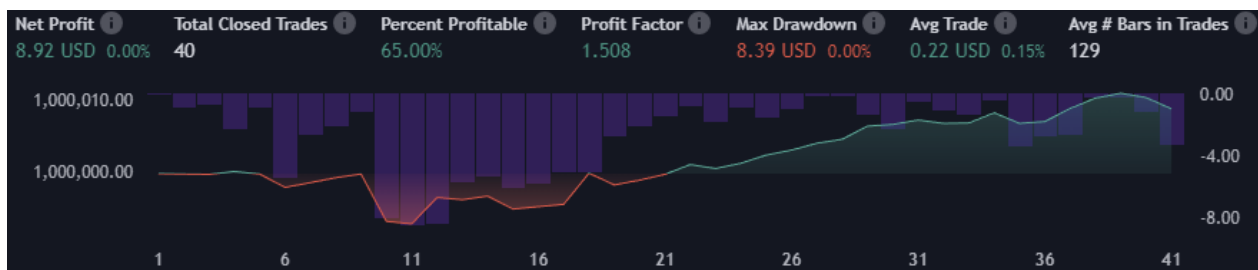


Fig. 5.2: Example of backtest results

5.5.2 Exceptional Scenarios

- The user selects incompatible instruments and indicators for the strategy. The tool will display an error then ask the user to retry with a different selection.
- The user enters invalid values as parameters for indicator calculations or for the strategy. The tool may generate a nonsensical strategy, or encounter an error then ask the user to retry with different inputs.

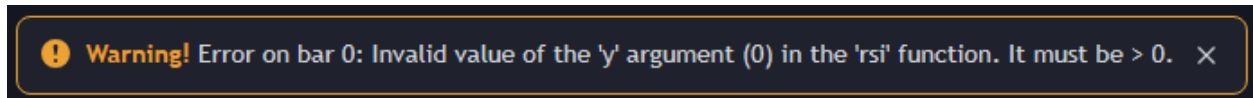


Fig. 5.3: An error message

6. Final Software Design

The final design of our software follows a standard client-server structure, with distinct frontend and backend servers that communicate with each other to fulfill user requests.

6.1 Frontend

The frontend server provides the interface that the users of our software interact with, and it is written in React, with Vite handling the building and deploying. The frontend's main responsibilities are retrieving pricing data from the backend, performing calculations required for the indicators, and then generating the appropriate graphs for the users. It is also responsible for accepting user inputs for backtest parameters, sending backtest requests to the backend, and retrieving backtest results to display to the user.

6.2 Backend

The backend server handles fetching asset pricing data from Yahoo Finance, caching data to improve performance, as well as running backtests on strategies as requested by the user through the frontend. The backend is a REST server written with Python using FastAPI and relies on the `yfinance.py` library for fetching pricing data, as well as the `backtest.py` library to backtest trading strategies. Pricing data is cleaned then passed directly to the frontend, whereas backtest results are first stored as an HTML file then passed to the frontend. There is also support for caching data using Redis, although it is optional, and all features will still work if Redis is not available.

6.3 Deployment

As of writing this document, the software is only available for self-hosting and is not yet deployed on the Internet. Those that want to use or host our software can clone the repository, then build and run the frontend and backend servers. However, due to the way we structured the software, it should also be relatively simple to deploy to any cloud platform. For deploying the frontend, Vite provides support for both static deployments and server-side rendered deployments. The backend server can also be easily deployed using either a virtual machine based or a serverless solution (albeit without caching).

7. Verification and Validation

We performed unit testing to validate and verify the correctness of our software's backend and frontend separately. For the backend, we created Python scripts to test the individual API endpoints and inspected the output to ensure the endpoints are working correctly. For example, we tested the data fetching endpoints to make sure the returned data is correct and properly formatted, and tested the backtesting endpoints to make sure the strategies are correctly executed. For the frontend, we relied on manual testing as we needed to make sure that a user's experience matches our designs and expectations. So, we tested each indicator with a variety of assets, then inspected the resulting graphs to ensure that they are acceptable. In addition, we used the frontend to test the backtesting features with various assets, to make sure that the resulting HTML files provided by the backend are properly loaded and contain the correct information. Lastly, we performed validation using real users with no prior experience during the symposium. We provided them with basic instructions on how to use the software, and they were able to successfully navigate between the various pages and utilize the features as intended.