Design and Development of a Quantitative Trading Platform

Data Visualisation, Portfolio Analysis and Optimisation, and Trading Simulation

Barnaby Todd

2025-10-01

Abstract

This project presents the design and implementation of a comprehensive quantitative trading application that integrates data acquisition, strategy backtesting, portfolio optimisation, and trading simulation within a unified environment. The platform enables users to retrieve and analyze historical market data via `yfinance`, construct and evaluate investment strategies across multiple asset classes, and perform rigorous backtesting using customizable hyperparameters and portfolio configurations. A multi-stage portfolio optimiser facilitates the construction of efficient portfolios based on user-defined universes and objectives, and the trading simulator provides a realistic environment for evaluating execution and performance outcomes. Together, these modules create an end-to-end workflow for quantitative research, empowering users to explore, test, and refine systematic trading approaches with analytical depth.

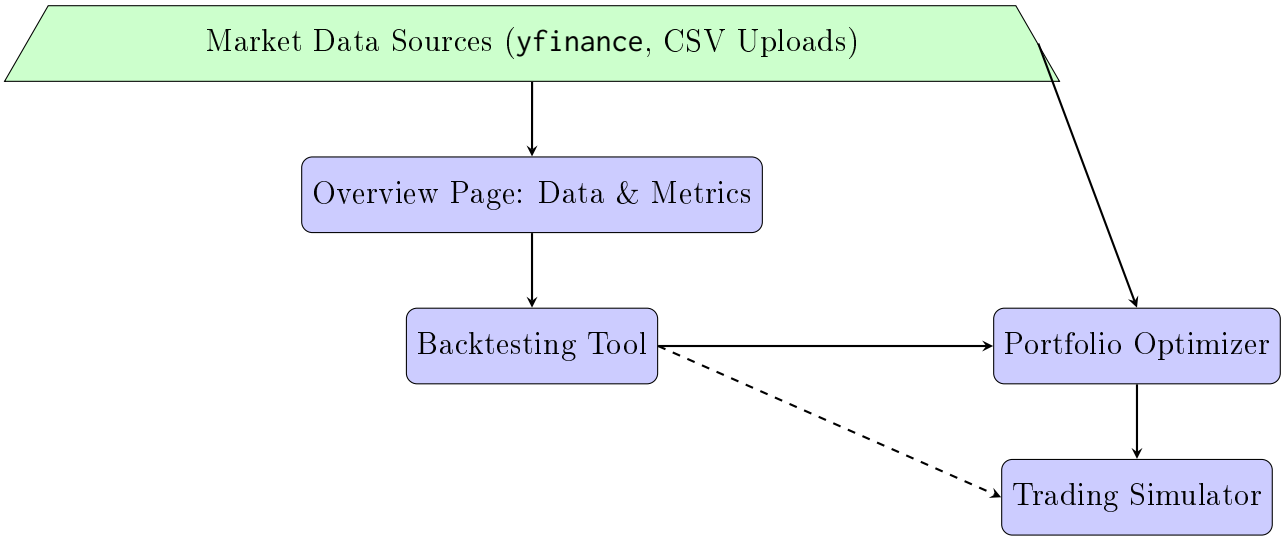
Table of Contents

1 Application and Software Design

This section discusses the system architecture, application modules, and software engineering techniques used to build the quantitative trading platform.

1.1 System Architecture

The system is designed as a modular, end-to-end quantitative trading platform. Data flows from market sources into analysis modules, through backtesting, portfolio optimization, and finally into a trading simulation environment. The architecture emphasizes modularity, allowing users to experiment with different strategies, hyperparameters, and portfolio configurations seamlessly.



Legend: Solid arrows indicate main data flow, while the dashed arrow represents inheritance of functionality.

1.2 Software Techniques and Design Patterns

1.2.1 Backend Design

The application uses a FastAPI backend, and adopts a modular monolithic backend architecture inspired by microservice principles. The following design choices have been used to maximise efficiency, scalability, and maintainability:

- **Modularity:** Each analytical module - such as data acquisition, backtesting, and portfolio optimization - is encapsulated in its own independent service within the codebase, maintaining its own logic, routes, and validation schemas.
- **Shared functionality:** Common database models and utility functions are abstracted into reusable libraries to minimize duplication.
- **Database abstraction:** All modules interact with a unified data layer built on Microsoft SQL Server, accessed via SQLAlchemy ORM for abstraction and schema management.
- **RESTful API design:** FastAPI exposes endpoints for strategy testing, portfolio optimization, simulation, and data management. Endpoints follow REST principles and are versioned for future extensibility.
- **Asynchronous and parallel processing:** Heavy quantitative tasks such as backtests and optimisation are executed asynchronously via Python's `asyncio` and concurrent executors to allow parallel computation.
- **Efficiency considerations:** Backtesting underpins many processes in the application, so performance optimizations are critical. Numerical computations are vectorized using libraries such as NumPy and pandas to avoid explicit Python loops. Additionally, walkforward backtests are structured to run each year individually and then combined into overlapping windows, ensuring that repeated calculations are avoided and computational resources are used efficiently.
- **Scalability and future architectural flexibility:** Although all modules currently operate within a single FastAPI instance connected to one database, the system's structure allows for the possibility of decoupling and deploying each component as an independent microservice in future iterations.

1.2.2 Frontend Design

The frontend of the application is implemented using the React JavaScript framework, providing a responsive and interactive interface for executing quantitative trading analyses. It enables users to manage datasets, run backtests, perform portfolio optimisation, and simulate trading outcomes through an intuitive graphical interface.

The design emphasises modularity, reusability, and real-time responsiveness, and employs Tailwind CSS for layout, to produce a clean and professional design. Each major feature is encapsulated within its own React component, which communicates with the backend via RESTful API endpoints exposed by the FastAPI server.

- **Component-based architecture:** The interface is structured around reusable, independent components such as navigation bars, data tables, charts, forms, and analysis result panels. Each component encapsulates its own state, logic, and presentation, allowing for scalable development and simplified debugging.
- **Data visualisation:** Quantitative results are rendered using dynamic charting libraries such as Recharts and Plotly. Users can visually analyse historical performance metrics, portfolio weights, and cumulative returns through interactive graphs and plots.
- **User interaction:** The interface provides interactive elements such as sliders, dropdown menus, and editable parameter cards, enabling users to customise input values and tailor application functionality to their objectives.
- **User feedback:** The interface provides real-time updates for computational tasks using Server-Sent Events (SSE) to stream progress. Loading indicators and progress bars keeping users informed of task completion and intermediate results.
- **Scalability and maintainability:** The component-based structure facilitates long-term maintainability. The modular codebase allows new analysis modules or visual components to be integrated with minimal refactoring.

1.3 Application Modules

The quantitative trading platform is divided into four main modules, each corresponding to a distinct page within the application. These modules are designed to interact seamlessly, allowing users to progress from data exploration to strategy analysis, portfolio optimization, and simulated trading. This modular approach ensures flexibility, maintainability, and a natural workflow for quantitative research.

1.3.1 Overview Page: Data and Metrics

The Overview module serves as the entry point to the platform. It allows users to retrieve historical data directly from `yfinance` to store in the application database. Once data is loaded, the system is designed to compute a variety of descriptive statistics and visualizations.

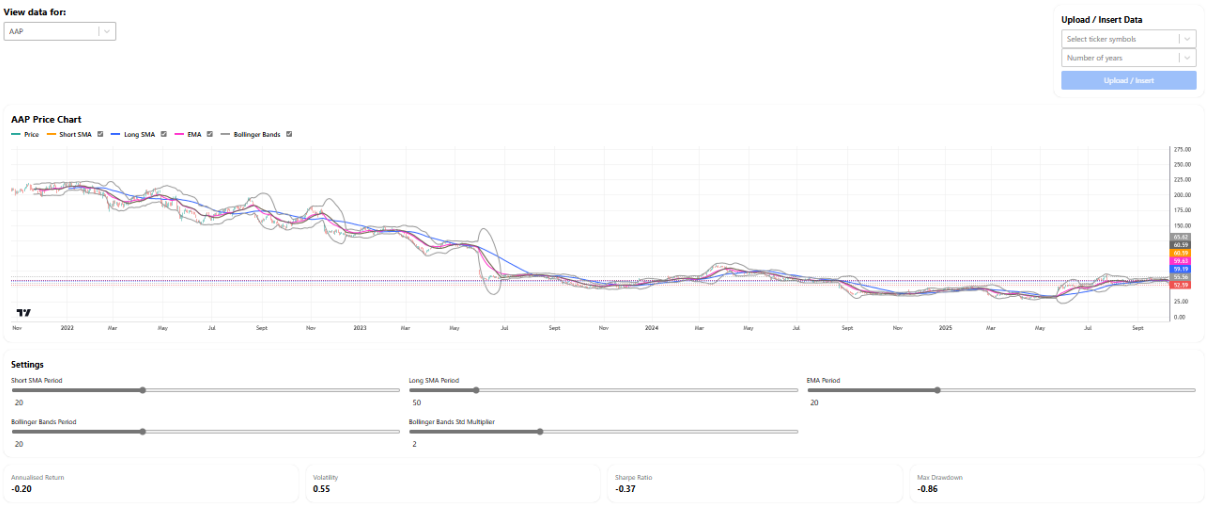


Figure 1: Application Dashboard

- **Data retrieval:** Supports automated fetching of stock or index data from online sources to upload to the database.
- **Visualization:** Interactive equity graphs with optional indicator overlays such as SMA and Bollinger Bands. Users can customize indicator parameters and display settings.
- **Metrics:** Summary statistics such as mean return, volatility, Sharpe ratio, and drawdown.

This page enables users to interactively visualize individual stock data, providing actionable insights into performance, and context while exploring the rest of the application.

1.3.2 Backtesting Tool

The Backtesting Tool enables users to design and evaluate trading strategies across multiple assets. It supports both single-strategy and blended portfolios using a modular configuration system.

- **Strategies:** Six built-in strategies - Momentum, Bollinger Band Reversion, RSI Reversion, Moving Average Crossover, Pairs Trading, and Breakout, with the option to create a customised portfolio using a combination of these strategies.

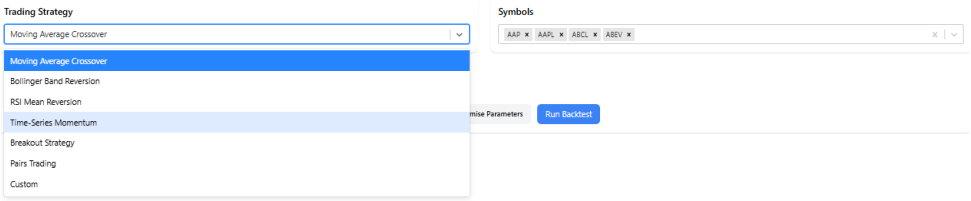


Figure 2: Backtesting Strategy Selection

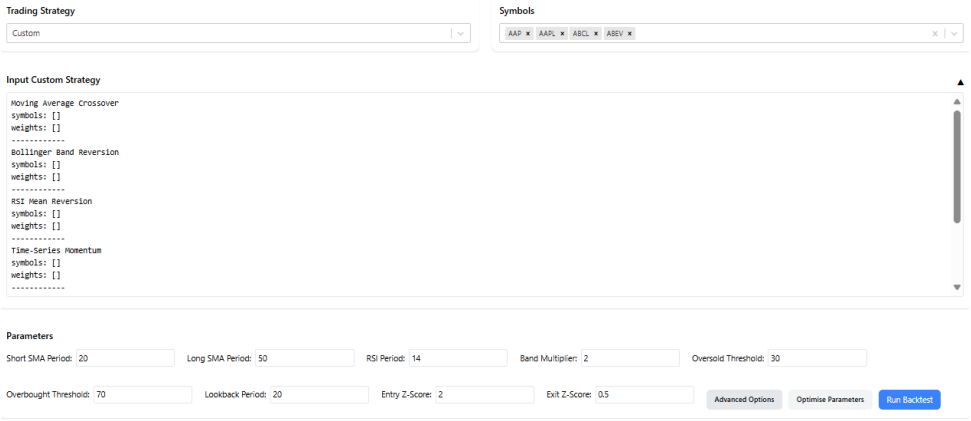


Figure 3: Backtesting Custom Strategy Input

- **Hyperparameters:** Users can configure strategy-specific parameters as well as global trading parameters such as transaction costs, rebalancing frequency and objectives, and portfolio weights.

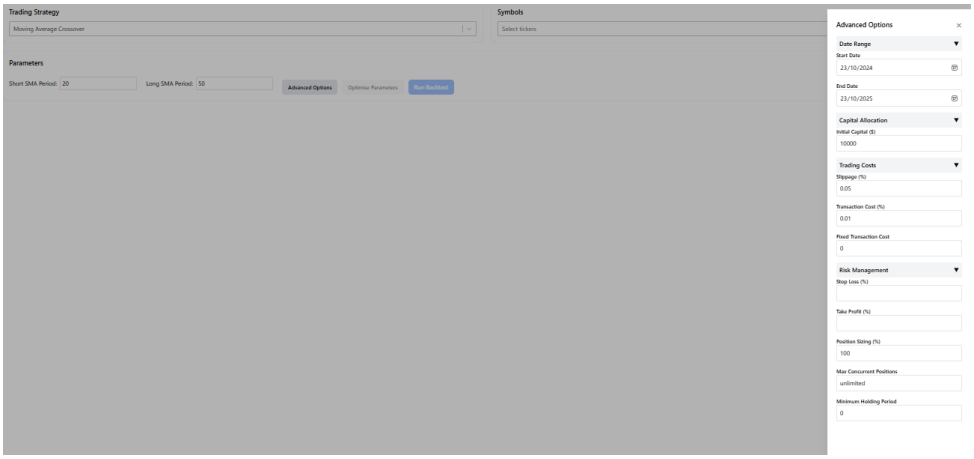


Figure 4: Backtesting Hyperparameter Configuration

- **Optimisation:** Strategy-specific hyperparameters can be optimised via Bayesian optimisation with cross validation. C-V fold length and the maximum number of optimisation iterations can be set by the user.

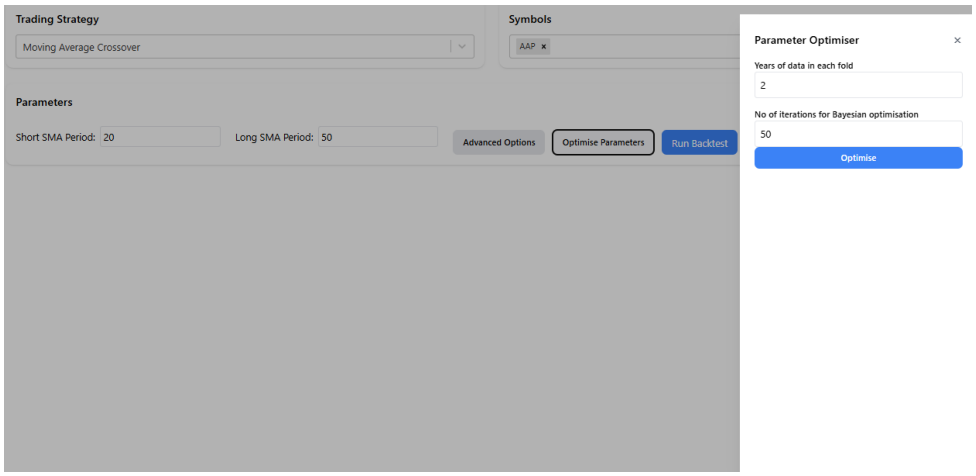


Figure 5: Backtesting Hyperparameter Optimisation

- **Pair selection:** For Pairs Trading, asset pairs are selected in the backend to maximise expected performance. Users receive all candidate pairs, with the optimal pairs pre-selected, enabling them to create a custom pair portfolio if desired.

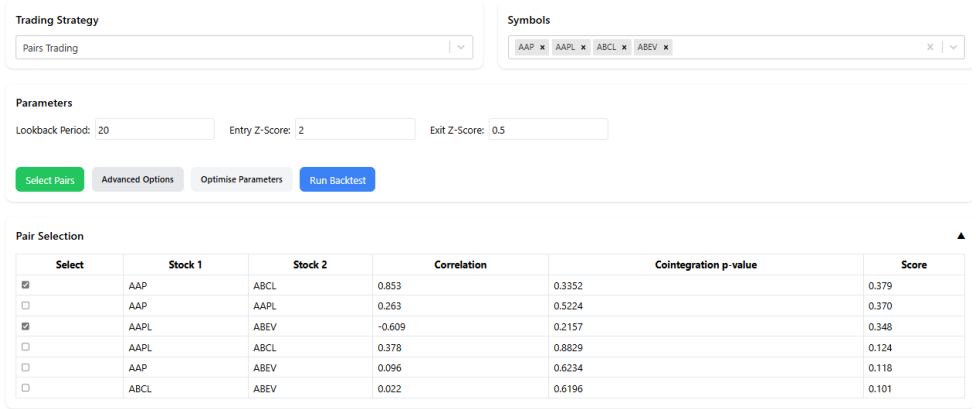


Figure 6: Backtesting Pair Selection

- **Outputs:** Equity curves, risk metrics, trade logs, and performance summaries.
- **Analysis:** Metrics such as CAGR, Sharpe ratio, max drawdown, and win rate are computed. Performance statistics and charts can be viewed across 3 different tabs - Overview, Trade Analytics and Risk & Exposure.

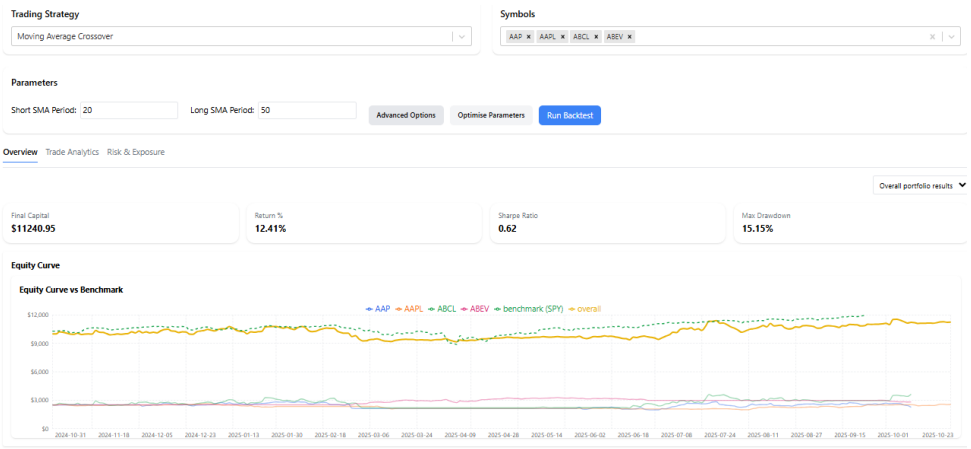


Figure 7: Backtesting Overview Tab

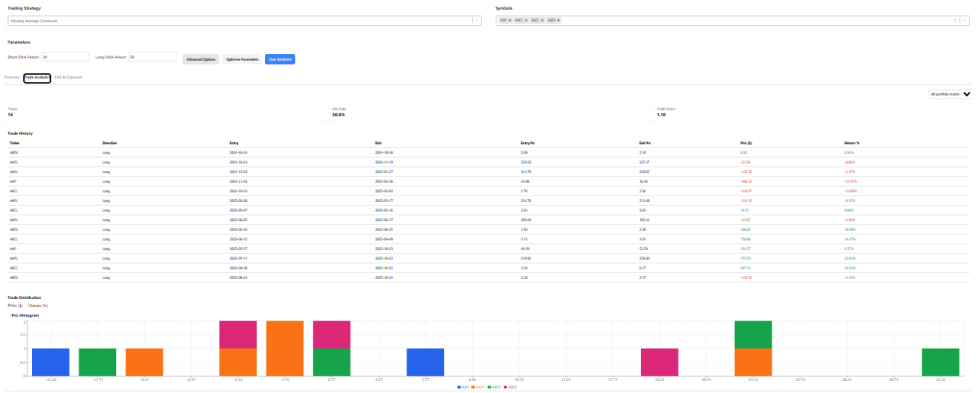


Figure 8: Backtesting Trade Analytics Tab

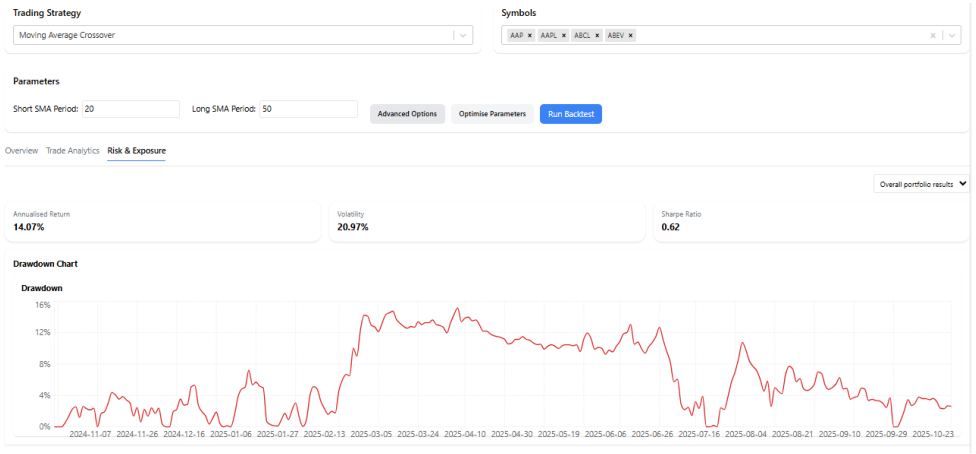


Figure 9: Backtesting Risk & Exposure Tab

1.3.3 Portfolio Optimizer

The Portfolio Optimizer implements a six-stage pipeline designed to generate the most efficient allocation from a given asset universe. Users can define constraints for the opti-

misation, and after each of the final 3 stages, the user can choose to save the generated portfolio to the database.

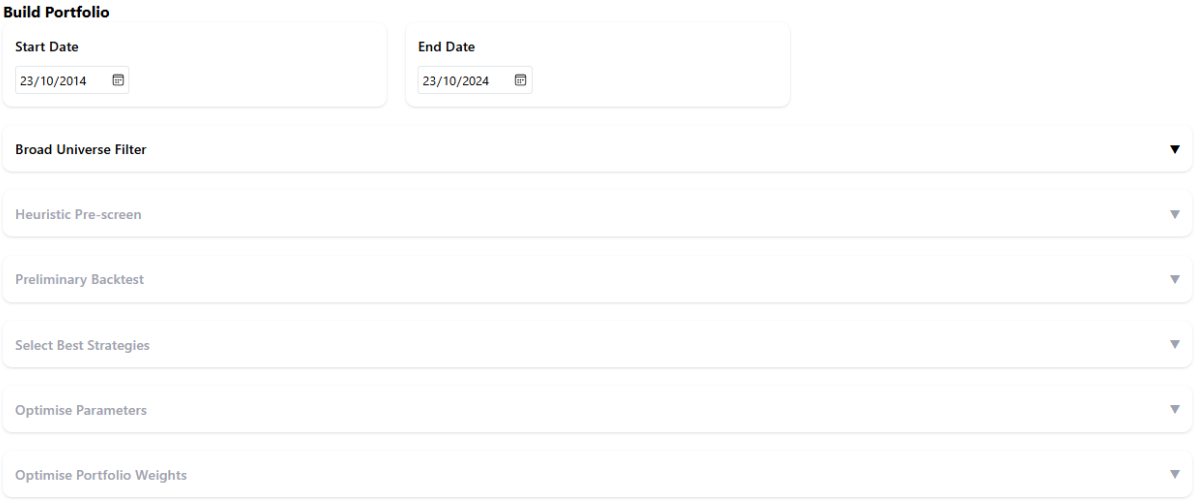


Figure 10: Portfolio Builder

1. **Initial universe filter:** The universe of assets available on `yfinance` is filtered via broad market and asset-level filters.

Figure 11: Portfolio Builder - Initial Universe Filter

2. **Pre-screening:** Assets are pre-screened using a number of tests to determine whether they are suitable for mean-reversion, momentum and breakout strategy types.

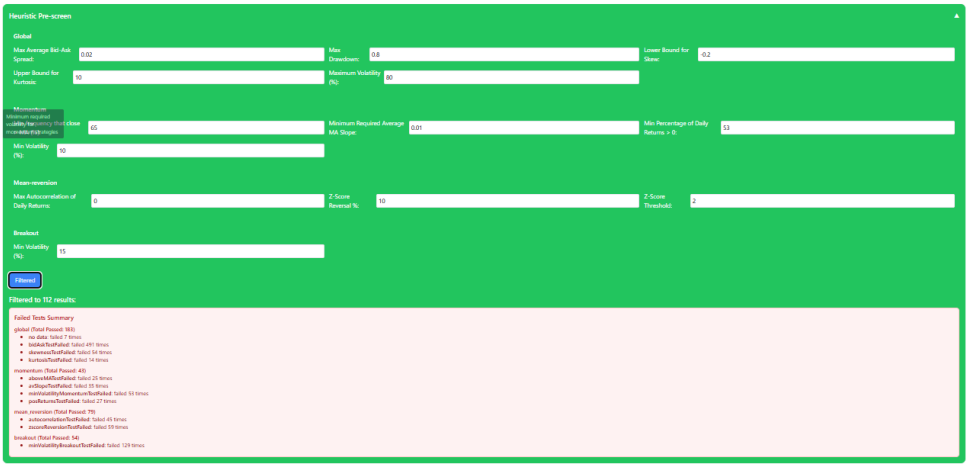


Figure 12: Portfolio Builder - Pre-screen

3. **Preliminary backtest:** Assets are run through a short backtest to ascertain which strategies have performance potential for each asset.

Figure 13: Portfolio Builder - Preliminary Backtest

4. **Strategy selection:** The best performing strategy for each asset is selected via a full walk-forward backtest, while filtering out any remaining under-performing assets.

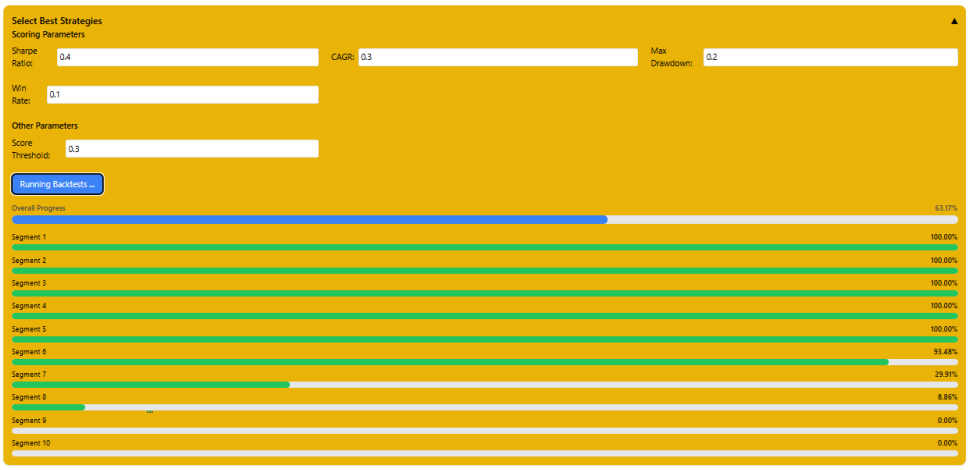


Figure 14: Portfolio Builder - Strategy Selector

5. **Hyperparameter optimisation:** The hyperparameters for each strategy are optimised using Bayesian optimisation techniques on the selected portfolio.

Figure 15: Portfolio Builder - Optimise Hyperparameters

6. **Optimise portfolio weights:** Weights are applied to the assets within the portfolio using Hierarchical Risk Parity allocation and constrained mean-variance optimisation.

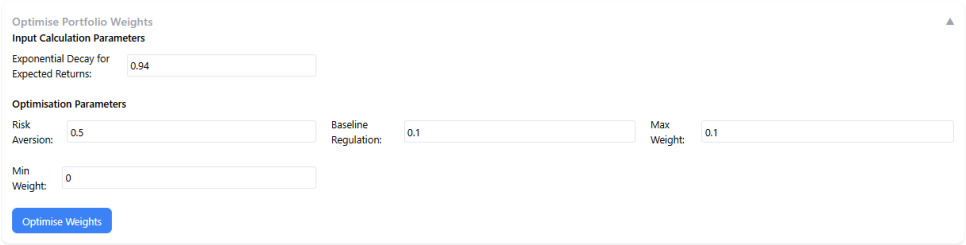


Figure 16: Portfolio Builder - Optimise Portfolio Weights

1.3.4 Trading Simulator

The Trading Simulator module provides a sandbox environment for simulating live trading scenarios using optimized or backtested portfolios. It reuses the logic found in the backtesting module.

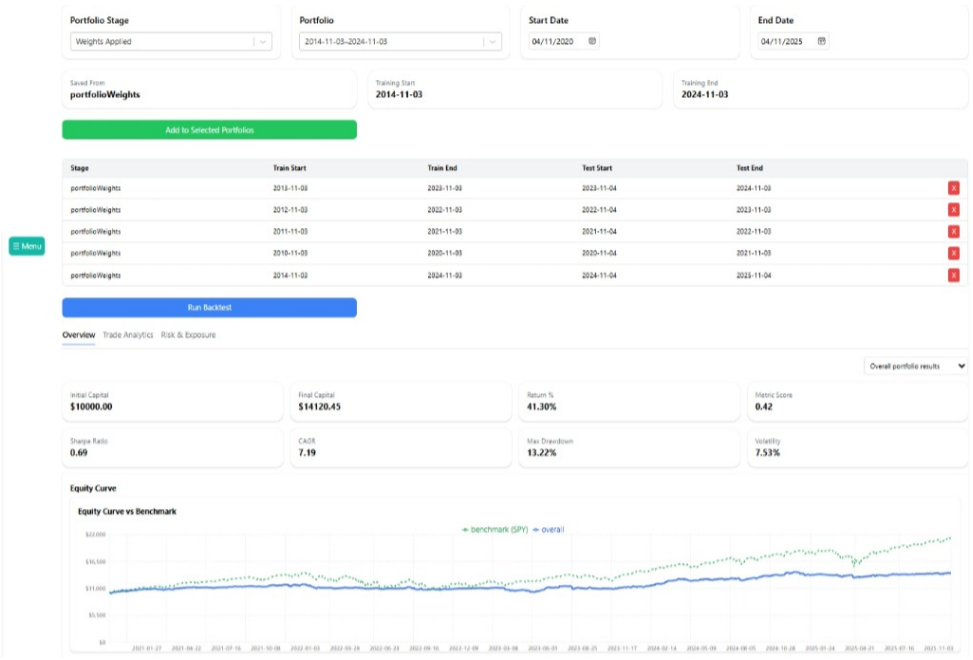


Figure 17: Trading Simulator

- **Simulation Engine:** Uses discrete time steps to emulate daily trading activity over a customisable date range.
- **Analysis:** Provides information on performance metrics, such as Sharpe ratio and return, trading analytics and risk measures

Together, these modules create a cohesive workflow: users begin with data exploration, develop and backtest trading strategies, optimize their portfolios, and finally simulate trading outcomes in a controlled environment.

2 Quantitative Methodology

This section describes the quantitative finance methodology underpinning the application.

2.1 Data Acquisition and Preprocessing

Data is obtained through the `yfinance` API, which provides historical price, volume, and fundamental information for global equities. Upon retrieval, the data undergoes preprocessing and cleaning to ensure consistency and integrity.

2.2 Backtesting Methodology

The backtesting framework simulates historical trading strategies to evaluate their performance. The process consists of generating signals, executing trades, and tracking portfolio evolution.

1. **Data loading:** Data is loaded for the specified date range, and includes a lookback window to ensure strategies have enough data to generate signals starting on the first trading date.

2. **Signal Generation:** Each strategy module defines entry and exit rules based on historical data.

3. **Execution Logic:** Positions are opened or closed according to generated signals, subject to user-defined parameters such as lookback periods, leverage, and transaction costs.

4. **Portfolio Updating:** Equity, position and available capital values are updated iteratively over time.

5. **Performance Evaluation:** Key metrics such as Compound Annual Growth Rate (CAGR), Sharpe Ratio, and Maximum Drawdown are computed after completion of the backtest process. Trade information, such as profit and loss, % return and execution dates, is recorded for each trade, as well as overall trade measures such as win rate and profit factor.

2.3 Strategy Design

Six baseline strategies are implemented in the platform, each representing a distinct class of trading behaviour. The modular design allows users to combine strategies or define custom ones.

- **Momentum:** Generates signals based on cumulative returns over a lookback window. Positive returns produce a buy signal and vice versa.

- **RSI Mean Reversion:** Generates signals based on a relative strength index score. RSI scores below the set oversold threshold produce a buy signal, whilst scores above the overbought threshold produce a sell signal.

- **Moving Average Crossover:** Generates signals when short-term averages cross long-term averages. Short-term average $>$ long-term average indicates momentum gain and generates a buy signal, and vice versa.

- **Pairs Trading:** Trades asset pairs based on deviations from their historical spread. When spread is unusually high, short the overperforming asset and long the underperforming asset. Exit when spread returns within a specified range. Pairs of assets are selected which historically move together, based on strong correlation and cointegration values. Maximum weight matching in weighted graphs is used to select the best overall pairs portfolio.

  1. Compute correlation coefficients ($corr_{ij}$) for each possible pair of assets i, j
  2. Use the Engle-Granger Cointegration Test to compute the cointegration p_{value}
  3. Compute each pair score using

$$\text{score}_{ij} = (w_{\text{corr}} \cdot |\text{corr}_{ij}| + w_{\text{coint}} \cdot (1 - p_{\text{value}})) * \beta_{pen}$$

     with $w_{\text{corr}}, w_{\text{coint}}$ as the weighting parameters, and β_{pen} as a penalty term for extreme β values during linear regression in the previous step.

  4. Complete the maximum weight matching problem by constructing a weighted undirected graph $G = (V, E)$:
     - $V = \{s_1, s_2, \ldots, s_N\}$ are the stocks
     - $E = \{(s_i, s_j)\}$ are edges weighted by the pair score $w_{ij} = \text{score}_{ij}$

     Then find a set of edges $M \subseteq E$ such that:

$$\max_{M} \sum_{(i,j) \in M} w_{ij} \quad \text{subject to} \quad \deg_M(s) \leq 1, \ \forall s \in V$$

- **Breakout Strategy:** Generates entry signals when price exits threshold, and exit signals when the price returns inside the threshold. The expectation is that the price continues on its directional move after exiting the threshold.

- **Bollinger Band Reversion:** Generates entry signals when price exits the Bollinger bands, and exit signals when it reverts back inside. The expectation is that the price will revert back inside the bands soon after exiting.

Each strategy is parameterized by a lookback period, and optional minimum holding duration, rebalance frequency, and stop-loss/take-profit thresholds among other adjustable parameters. This enables extensive experimentation and hyperparameter tuning.

2.4 Portfolio Optimization Framework

The optimization module is based on modern portfolio theory (MPT) and its extensions. It aims to select the optimal portfolio from a universe of assets and apply portfolio weightings that optimize specific objectives under defined constraints. It executes a 6-stage pipeline to achieve this. Users can choose the date range over which to train the portfolio.

2.4.1 Initial Universe Filter

The asset universe is filtered based on customisable parameters. The user can select:

- regions
- exchanges
- sectors
- industries

to define the search area. They can also apply thresholds for:

- minimum average daily volume
- minimum market cap
- minimum earnings per share growth
- maximum price/earnings ratio

This stage applies these filters using built-in `yfinance` filters.

2.4.2 Heuristic Pre-screening

Assets are pre-screened to determine their suitability for momentum, mean-reversion and breakout strategies based on a series of tests. These tests are run over both a 1 year and

3 year period to determine recent, and longer term suitability. The assets must pass in at least one of the periods. The user can set the thresholds for each of the tests.

First, the assets are passed through global tests to ensure they are suitable for any trading strategy:

- **Bid-ask test:** Checks if the average bid-ask spread is below the set threshold to ensure suitable market liquidity and realistic trade execution
- **Max drawdown test:** Checks if the max drawdown for each asset is below the set threshold to filter out overly volatile and risky assets
- **Skewness test:** Checks if skewness is above the set threshold to reduce exposure to extreme losses
- **Kurtosis test:** Checks if kurtosis is below the set threshold to reduce exposure to extreme losses
- **Volatility test:** Checks if annualised volatility is below the set threshold to filter out overly volatile and risky assets

If assets pass the global tests, they are then passed through strategy group specific tests to determine suitability for the discussed strategy groups.

To determine suitability for momentum strategies, the assets are passed through the following tests:

- **Above MA test:** Calculates the percentage of data entries for which the asset price is above its 200-day moving average. The percentage must be above the set threshold to pass. This ensures the asset displays persistent trends.
- **Average slope test:** Calculates the average slope between each price and the previous price. The average slope must be above the set threshold to pass. This ensures consistent and strong trend movement.
- **Positive returns test:** Checks whether the percentage of daily returns which are positive exceeds the set threshold. This ensures consistent trends.
- **Volatility test:** Asset volatility must exceed the set threshold to filter out stagnant assets.

To determine suitability for mean-reversion strategies, the assets are passed through the following tests:

- **Autocorrelation test:** Checks if asset autocorrelation is below the set threshold, ensuring that, on average, price increases indicate a pending decrease and vice versa.
- **z-score reversion test:** Checks how often prices revert after a z-score deviation. Users can set the threshold which indicates a deviation and the minimum required reversion percentage. This checks that prices consistently revert towards the mean after deviating.

To determine suitability for breakout strategies, the assets are passed through the following tests:

- **Volatility test:** Checks that annualised volatility exceeds threshold to ensure price movements are large enough to produce meaningful breakouts

Selected asset-strategy group pairings are passed on to the next stage.

2.4.3 Preliminary Backtest

Assets are passed through a short preliminary backtest over the most recent year in the training period. This tests all strategies within the selected strategy groups for each stock and filters out strategies which underperform. Users can choose a Sharpe ratio threshold which tests must exceed to pass this stage. Asset-strategy pairs which pass this stage are then sent on to the next stage.

2.4.4 Strategy Selector

The final portfolio should only contain 1 strategy per asset. The purpose of this stage is to select the best strategy for each asset. Asset-strategy pairs are submitted to a full walk-forward backtest over the whole training period.

1. A backtest is run for each year in the training window in parallel.
2. The backtest results are processed to produce results for the walk-forward windows, by scaling the equity curves for each year in the window to continue from the final capital of the previous year

3. The walk-forward results are aggregated using the mean score of metric results.

A score is then calculated for each asset-strategy pair using a weighted sum of normalised values for Sharpe ratio, compound annual growth rate, max drawdown and win rate, with the user choosing the weights for each metric. The highest scoring strategy for each asset is selected. It must exceed a user-specified minimum score to be passed on to the next stage.

2.4.5 Parameter Optimisation

Strategy-specific hyperparameters are optimised in this stage using Bayesian optimisation. Users can specify the scoring weights as in the previous stage and the maximum number of trials in the optimisation process.

Here, Optuna is used as the optimisation framework, which uses a **Tree-structured Parzen Estimator (TPE)** approach to model the probability distributions of good and bad parameter regions:

1. **Split prior observations:** At iteration t, the set of completed trials is divided into:
$$L = \{\boldsymbol{\theta}_i \mid f(\boldsymbol{\theta}_i) \geq y^*\}, \quad G = \{\boldsymbol{\theta}_i \mid f(\boldsymbol{\theta}_i) < y^*\},$$
where y^* is a quantile threshold (e.g. the top 15%).

2. **Model likelihoods:** Optuna estimates two density models:
$$l(\boldsymbol{\theta}) = p(\boldsymbol{\theta} \mid f(\boldsymbol{\theta}) \geq y^*), \quad g(\boldsymbol{\theta}) = p(\boldsymbol{\theta} \mid f(\boldsymbol{\theta}) < y^*).$$

3. **Sampling rule:** New candidates are sampled by maximizing the ratio:
$$\boldsymbol{\theta}_{t+1} = \arg\max_{\boldsymbol{\theta}} \frac{l(\boldsymbol{\theta})}{g(\boldsymbol{\theta})},$$
which biases the search toward regions likely to yield higher scores.

2.4.6 Portfolio Weight Optimisation

This stage is run via a 3-stage pipeline.

1. **Input calculations:** Covariance and expected return matrices are computed for the selected portfolio.

   - **Expected returns:** The expected return for each asset is estimated using an exponentially weighted moving average (EWMA) of historical returns:
   - **Risk (covariance) matrix:** The covariance between assets i and j is computed as:

   $$\Sigma_{ij} = \frac{1}{T-1}\sum_{t=1}^{T}(r_{t,i} - \bar{r}_i)(r_{t,j} - \bar{r}_j),$$

   where $r_{t,k}$ is the return value for asset k at time t and $\bar{r}_k$ is the sample mean of asset k's returns.

2. **HRP allocation:** Weights are allocated via hierarchical risk parity allocation.

   - **Step 1: Correlation distance matrix** Convert asset correlations into distances:

   $$D_{ij} = \sqrt{\frac{1-\rho_{ij}}{2}},$$

   where ρ_{ij} is the correlation between assets i and j. Smaller distances indicate higher correlation.

   - **Step 2: Hierarchical clustering** Apply single linkage to the distance matrix D. This recursively merges clusters of assets (or individual assets into a cluster) by merging the clusters with the smallest minimum distance between them at each step. This produces a tree of merges (dendrogram) that defines the nested cluster structure.
   - **Step 3: Recursive bisection weighting** At each split of the hierarchy in the dendrogram, weights are assigned recursively to ensure **equal risk contribution** across sub-clusters:

   $$w_A = \frac{1/\sigma_A^2}{1/\sigma_A^2 + 1/\sigma_B^2}, \quad w_B = 1 - w_A,$$

   where $\sigma_A^2 = \mathbf{w}_A^\top \Sigma_A \mathbf{w}_A$ and Σ_A is the covariance submatrix for cluster A.

   The recursion continues until single-asset clusters are reached, yielding the final HRP weight vector $\mathbf{w}_{HRP}$.

3. **Mean-Variance Portfolio Optimisation**

   The final weights are obtained by solving a constrained quadratic optimisation problem:

$$\begin{aligned} \max_{\mathbf{w}} \quad & \mathbf{w}^\top \mu - \frac{\gamma}{2} \mathbf{w}^\top \Sigma \mathbf{w} - \lambda \|\mathbf{w} - \mathbf{w}_0\|_2^2 \\ \text{s.t.} \quad & \sum_{i=1}^{N} w_i = 1, \\ & w_{\min} \leq w_i \leq w_{\max}, \quad \forall i, \end{aligned}$$

   where:

   - $\mathbf{w}$: vector of portfolio weights
   - μ: expected returns vector
   - Σ: covariance matrix
   - γ: risk-aversion coefficient
   - λ: baseline regularization strength
   - $\mathbf{w}_0$: baseline (prior) weight vector
   - $w_{\min}, w_{\max}$: weight constraints per asset

   The term $\lambda\|\mathbf{w} - \mathbf{w}_0\|_2^2$ penalizes deviations from a baseline portfolio, encouraging stability and reducing overfitting. Using the HRP weights as $\mathbf{w}_0$ ensures that the resulting portfolio remains diversified while incorporating mean-variance preferences.

   This is implemented using `cvxpy` with the additional constraints that $\sum_i w_i = 1$ and $min_{weight} < w_i < max_{weight}\ \forall i$

3 Results and Analysis

This section presents empirical results produced by the quantitative trading platform. The objective is to evaluate the effectiveness of the implemented strategies, and optimization framework, under realistic market conditions. The analysis focuses on portfolio performance, risk characteristics, and comparative results across different configurations.

3.1 Experimental Setup

The system was tested on a selection of U.S. equities across multiple sectors obtained from the `yfinance` API. Daily price data was collected for a 15-year period (2010–2025). All backtests assume a transaction cost of 0.01% and 0.05% slippage per trade unless otherwise specified.

- **Universe:** 750 medium-to-large-cap U.S. stocks
- **Benchmark:** S&P 500 Index
- **Risk-free rate:** 2.0% annualized
- **Initial capital:** $10,000

5 Portfolios were built using 10-year training periods ranging from 2010-2020 to 2014-2024.

3.2 Results

The following results show the yearly metric values for the different portfolios. Overall scores are computed using

$$score = 0.5 * \tilde{s} + 0.3 * \tilde{c} + 0.2 * \tilde{d}_{max}$$

where:
$\tilde{s}$ = normalised Sharpe Ratio
$\tilde{c}$ = normalised CAGR and
$\tilde{d}_{max}$ = 1− normalised Max Drawdown

3.2.1 Comparative Results with S&P 500 Benchmark

Figure 18 illustrates the cumulative equity curve generated by using each portfolio for exactly 1 year after the end of its training period. The S&P 500 equity curve is shown alongside for comparison.

Table 1 summarizes key risk and performance metrics for each portfolio .

We can draw several key observations from these results:

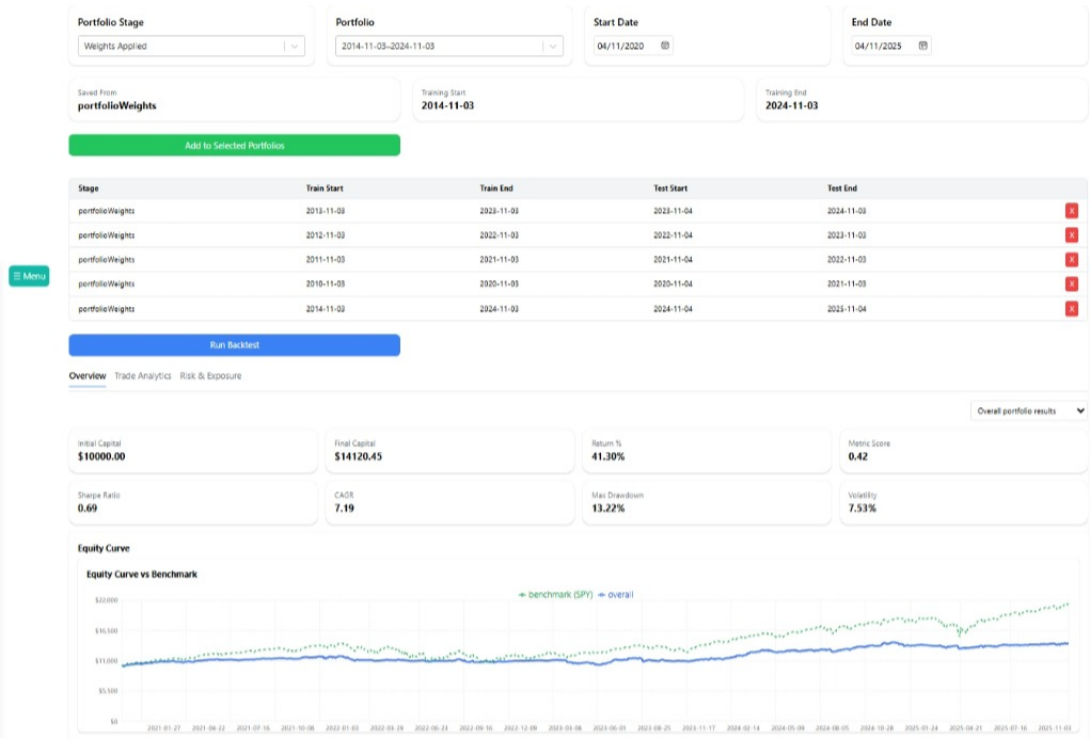


Figure 18: Cumulative equity curves for selected strategies (2015–2025).

Portfolio [Training Dates] (No. of Assets)	**Metric**	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	Train Av.	Test Av.
Portfolio 1 [2010-2020] ()	CAGR	0.26	2.35	13.55	6.25	11.21	3.80	15.76	7.08	12.12	11.26	15.94	-7.62	1.70	8.09	-2.06	8.36	3.21
	Sharpe	-0.22	0.08	2.02	0.86	1.52	0.37	2.73	0.80	1.44	0.83	2.18	-1.79	-0.01	0.96	-0.63	1.04	0.14
	Max Drawdown (%)	9.33	7.21	3.01	3.76	2.63	3.01	1.44	4.37	5.81	11.39	3.12	10.14	5.89	5.29	7.23	5.20	6.33
	Volatility (%)	6.84	6.36	5.39	4.87	5.77	5.06	4.66	6.31	6.72	11.22	5.95	5.45	6.69	6.23	6.14	6.32	6.09
	Overall Score	0.41	0.47	0.75	0.59	0.69	0.53	0.85	0.58	0.67	0.56	0.78	0.20	0.46	0.60	0.37	0.61	0.48
Portfolio 2 [2011-2021] (56)	CAGR	-	5.51	9.55	5.83	7.42	0.56	13.54	5.81	8.55	4.52	26.06	-5.95	0.16	8.95	-1.70	8.74	0.37
	Sharpe	-	0.56	1.30	0.78	0.81	-0.18	2.33	0.55	0.96	0.28	3.09	-1.00	-0.21	1.13	-0.67	1.05	-0.19
	Max Drawdown	-	5.03	3.20	2.96	5.15	4.82	1.94	4.79	4.16	11.28	3.78	9.64	5.29	3.56	4.54	4.71	5.76
	Volatility	-	6.38	5.59	4.87	6.61	6.82	4.64	6.68	7.03	10.85	6.94	7.82	7.48	5.96	5.30	6.64	6.64
	Overall Score	-	0.55	0.65	0.58	0.58	0.44	0.79	0.55	0.61	0.47	0.92	0.30	0.44	0.63	0.38	0.62	0.44
Portfolio 3 [2012-2022] (30)	CAGR	-	-	6.58	7.21	-0.27	-0.48	4.12	4.25	8.91	4.70	12.45	19.91	1.49	8.47	2.34	6.74	4.10
	Sharpe	-	-	0.70	1.05	-0.43	-0.35	0.46	0.43	1.38	0.36	1.57	3.00	-0.03	1.01	0.08	0.82	0.35
	Max Drawdown	-	-	4.08	3.62	4.02	6.09	2.96	5.54	2.06	6.48	3.70	1.57	7.98	4.17	3.22	4.01	5.12
	Volatility	-	-	6.55	4.82	5.03	6.50	4.65	5.39	4.81	8.21	6.33	5.43	7.85	6.28	6.74	5.77	6.96
	Overall Score	-	-	0.57	0.61	0.42	0.41	0.54	0.52	0.67	0.51	0.69	0.89	0.45	0.61	0.49	0.58	0.52
Portfolio 4 [2013-2023] (33)	CAGR	-	-	-	11.25	7.87	3.83	19.95	14.11	13.31	10.90	20.58	-2.90	17.65	25.89	8.63	11.66	17.26
	Sharpe	-	-	-	1.18	0.64	0.29	2.72	1.26	1.11	0.63	1.97	-0.49	1.67	2.73	0.80	1.10	1.77
	Max Drawdown	-	-	-	4.96	6.26	5.98	2.55	6.79	8.29	14.76	4.94	10.71	4.20	3.63	5.61	6.94	4.62
	Volatility	-	-	-	7.62	9.43	6.97	6.03	9.21	9.94	15.01	8.68	9.20	8.79	7.83	8.26	9.09	8.05
	Overall Score	-	-	-	0.64	0.56	0.50	0.86	0.65	0.62	0.52	0.77	0.36	0.73	0.89	0.58	0.62	0.73
Portfolio 5 [2014-2024] (52)	CAGR	-	-	-	-	4.47	5.31	13.65	4.94	4.00	-1.10	21.69	-3.17	6.27	28.78	3.00	8.48	3.00
	Sharpe	-	-	-	-	0.39	0.59	2.43	0.49	0.34	-0.23	2.22	-0.62	0.65	3.07	0.16	0.93	0.16
	Max Drawdown	-	-	-	-	4.87	3.03	1.99	5.25	8.03	11.75	3.75	7.14	4.76	4.04	6.94	5.46	8.30
	Volatility	-	-	-	-	6.62	5.64	4.49	6.11	6.15	10.95	8.09	7.96	6.59	7.69	6.88	7.03	7.96
	Overall Score	-	-	-	-	0.52	0.56	0.80	0.53	0.50	0.39	0.81	0.37	0.56	0.93	0.47	0.60	0.47
Combined Portfolio	CAGR	-	-	-	-	-	-	-	-	-	-	15.94	-5.95	1.49	25.89	3.00	-	8.07
	Sharpe	-	-	-	-	-	-	-	-	-	-	2.18	-1.00	-0.03	2.73	0.16	-	0.81
	Max Drawdown (%)	-	-	-	-	-	-	-	-	-	-	3.12	9.64	7.98	3.63	8.30	-	6.53
	Volatility (%)	-	-	-	-	-	-	-	-	-	-	5.95	7.82	7.85	7.83	7.96	-	7.48
	Overall Score	-	-	-	-	-	-	-	-	-	-	0.78	0.30	0.45	0.89	0.47	-	0.58
Benchmark	CAGR	4.59	15.36	27.43	15.85	6.67	1.21	26.48	6.95	14.72	13.80	37.74	-18.22	17.40	32.77	21.49	-	14.95
	Sharpe	0.22	0.89	1.97	1.22	0.37	0.01	3.01	0.40	0.82	0.50	2.50	-0.84	0.99	2.25	0.99	-	1.02
	Max Drawdown	18.61	9.69	5.55	7.27	11.91	12.69	2.61	10.10	16.12	33.72	5.11	24.50	9.97	8.41	18.76	-	13.00
	Volatility	21.41	15.00	11.67	10.88	14.87	13.90	7.23	14.16	15.74	33.10	12.33	23.21	15.40	12.02	19.62	-	16.04
	Overall Score	0.42	0.60	0.80	0.65	0.49	0.42	0.92	0.51	0.55	0.41	0.92	0.15	0.62	0.84	0.59	-	0.59

Table 1: Performance metrics of individual portfolios (2011–2025). The Combined portfolio is created by taking the first test year of each of the other portfolios.

- **Risk reduction:** Both volatility and maximum drawdown are substantially lower - by approximately 40-50% - in all model portfolios compared with the benchmark.

Moreover, these risk metrics remain relatively stable between the training and testing periods, indicating that the model has effectively stabilised portfolio behaviour and achieved consistent risk reduction.

- **Performance consistency:** The portfolios exhibit more stable performance over time, with fewer large fluctuations than the benchmark. Notably, all five portfolios outperformed the benchmark during the 2022 market downturn, demonstrating greater resilience to adverse market regimes.

- **Test performance:** The combined portfolio achieves risk-adjusted scores comparable to the benchmark, while absolute returns are generally lower. However, this comes with a marked decrease in volatility and drawdowns. The portfolio performed notably better than the benchmark during stress periods (e.g., 2022), though it underperformed during strong recovery phases such as 2023 and 2025.

Figure 19 provides a clearer view of the evolution of portfolio performance relative to the benchmark over time.

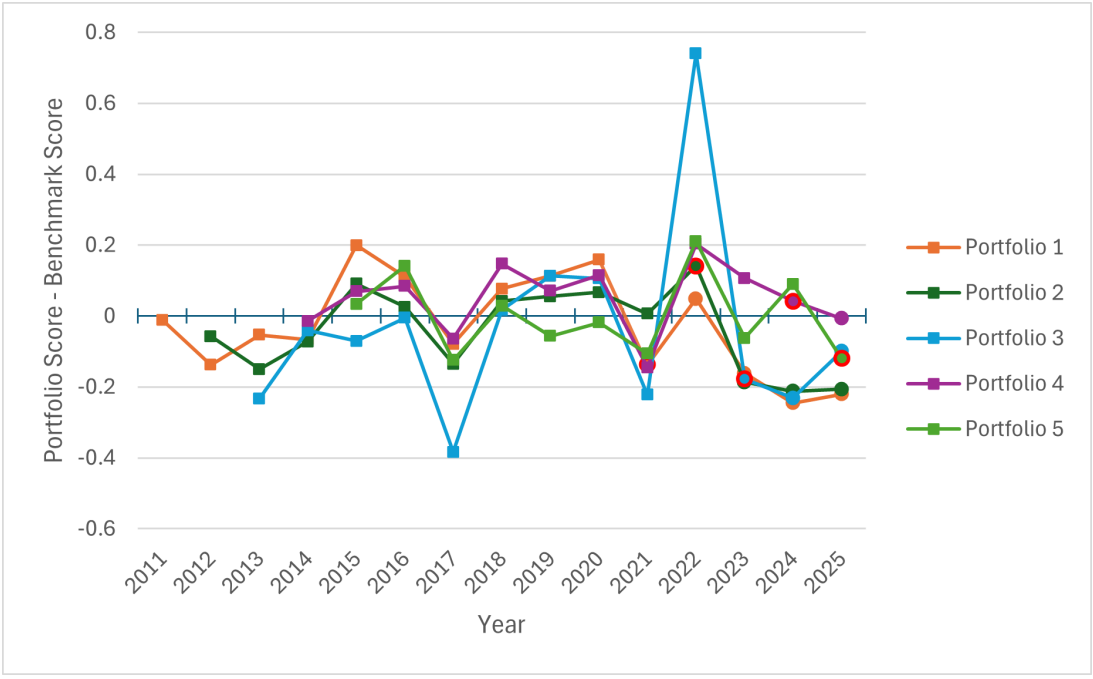


Figure 19: Net Portfolio Scores in Excess of Benchmark Scores

Several further insights can be drawn from this figure:

- **Diminishing out-of-sample performance:** Test scores tend to decline as the temporal distance from the training period increases. This behaviour is expected, as

older training data becomes less representative of the prevailing market conditions.

- **Initial out-of-sample generalisation:** The first out-of-sample results (circled in red) for Portfolios 1, 2, and 4 closely match the training scores of the other portfolios, suggesting that these models generalised well and avoided overfitting. In contrast, Portfolios 3 and 5 recorded weaker first out-of-sample results, indicating potential difficulty in adapting to market regime shifts in those years.

- **Market regime dynamics:** Portfolio performance often moves inversely to benchmark performance, reflecting the model's design to prioritise volatility reduction and downside protection. Consequently, the portfolios display reduced sensitivity to regime changes. Portfolio 3 exhibits this effect most prominently, likely because its final training year (2022) coincided with a major market decline, biasing its strategy selection toward assets and approaches that perform best in defensive or bearish environments.

3.2.2 Portfolio Optimization Outcomes

Table 2 displays the results for each of the portfolios discussed above after each of the final 3 stages of portfolio optimisation. Here, train results are the results generated by completing 1 backtest over the whole training period, and test results are the results generated in the first year after the training period.

Portfolio	Metric	Strategy Select		Param Optimisation		Portfolio Weights	
		Train	Test	Train	Test	Train	Test
1	Sharpe	1.04	0.83	0.97	1.59	0.90	2.18
	CAGR	9.63	6.74	10.95	14.35	8.93	15.94
	Max Drawdown	9.25	3.90	16.16	4.73	16.29	3.12
	Volatility	7.18	5.67	9.06	7.36	7.61	5.95
	Score	0.59	0.59	0.55	0.70	0.53	0.78
2	Sharpe	1.19	-1.72	1.19	-1.51	0.85	-1.00
	CAGR	12.21	-11.85	14.65	-10.37	8.20	-5.95
	Max Drawdown	13.93	14.03	20.67	13.76	12.50	9.64
	Volatility	8.29	8.31	10.25	8.33	7.22	7.82
	Score	0.59	0.16	0.56	0.19	0.54	0.30
3	Sharpe	0.92	0.81	0.90	0.62	0.79	-0.03
	CAGR	7.08	5.76	7.39	4.99	7.32	1.49
	Max Drawdown	6.44	4.06	6.74	3.20	10.12	7.98
	Volatility	5.45	4.59	5.91	4.81	6.66	7.85
	Score	0.58	0.58	0.58	0.56	0.55	0.45
4	Sharpe	0.89	1.51	0.85	1.85	1.10	2.73
	CAGR	9.55	12.33	9.32	16.05	12.94	25.89
	Max Drawdown	12.18	4.12	15.24	3.33	14.44	3.63
	Volatility	8.37	6.54	8.52	7.10	9.66	8.26
	Score	0.56	0.69	0.53	0.74	0.58	0.89
5	Sharpe	0.83	0.59	0.96	0.51	0.82	0.39
	CAGR	8.82	6.02	10.15	5.53	8.68	4.58
	Max Drawdown	10.94	6.56	13.10	7.29	11.13	6.94
	Volatility	8.20	6.90	8.31	7.08	8.15	6.88
	Score	0.55	0.54	0.56	0.53	0.55	0.51

Table 2: Performance metrics and computed Overall Scores for each portfolio at each stage of optimisation.

From these results, several key insights can be drawn:

- **Portfolio 1:** Demonstrated strong generalisation, with each stage contributing to improved out-of-sample performance as intended.
- **Portfolio 2:** Despite a decline in training performance, test results improved in the final stage. This indicates that the diversification introduced through HRP

weight optimisation effectively controlled risk and mitigated potential losses during an adverse market regime.

- **Portfolio 3:** Performance deteriorated markedly following the final portfolio weight optimisation. This likely reflects the highly concentrated market environment of 2023, in which technology and communication sectors dominated returns. The enforced diversification under HRP allocation prevented the portfolio from fully capturing this sectoral outperformance. Additionally, as HRP expected returns rely heavily on recent data, using 2022 returns - a year of broad market weakness - likely provided poor forward-looking estimates for 2023. Furthermore, asset selection was also based on 2022 performance, which may have resulted in a suboptimal subset for 2023.

- **Portfolio 4:** Achieved strong results, with substantial performance gains during the final weight optimisation stage. Given that asset selection and expected return estimation were heavily influenced by 2023 data, and that the 2024 market regime was broadly similar, the HRP optimisation benefited from a favourable and consistent input environment.

- **Portfolio 5:** Delivered stable but unremarkable results, with limited improvement across stages. This may reflect a regime shift between 2024 and 2025, as assets selected based on 2024 performance were not well-suited to the changing conditions of 2025.

Overall, the results indicate that the portfolio optimisation module is capable of constructing robust portfolios that can consistently outperform benchmark portfolios during periods of market stability. However, during periods of heightened volatility or regime shifts, the portfolios exhibit weaker generalisation and reduced performance consistency.

4 Conclusion and Future Work

4.1 Conclusion

This project presented the design and implementation of a comprehensive quantitative trading platform that integrates data acquisition, strategy development, backtesting, port-

folio optimization, and trading simulation within a single framework. The system was engineered to provide an end-to-end workflow for quantitative research, enabling users to retrieve market data, construct and evaluate investment strategies, and assess portfolio performance under realistic trading conditions.

Through modular design and a clear separation of functionality, the platform facilitates experimentation and comparative analysis across different strategies and portfolio configurations. The results demonstrate that the portfolio optimisation module can generate superior risk-adjusted returns relative to naive benchmarks, and that the trading simulator can be used to validate the robustness of portfolios under simulated execution environments.

Overall, the system successfully achieves its objective of providing a unified, flexible, and transparent toolset for quantitative finance research. It bridges the gap between theoretical model development and practical portfolio management, offering both analytical insight and operational depth.

4.2 Current Limitations

While the platform provides extensive analytical functionality, several limitations remain:

- The backtesting engine assumes daily data and does not yet account for intraday price dynamics or high-frequency trading effects.
- Backtesting operates using a limited number of basic hyperparameters on a limited number of strategies, which are mostly set up only for long-only exposure. Higher complexity portfolios and models are not yet supported.
- Market frictions such as liquidity constraints, bid-ask spreads, and slippage are modeled in a simplified manner.
- The optimization module currently focuses on classical mean-variance frameworks; advanced risk models (e.g., CVaR, Black-Litterman) are not yet integrated.
- The trading simulator operates in a controlled environment without connection to real brokerage APIs or live data feeds.

These limitations represent opportunities for continued refinement and expansion of the system.

4.3 Future Work

Immediate development efforts will focus on enhancing the stability and adaptability of the portfolio optimiser during periods of market volatility or regime transitions. The following improvements are proposed:

- **Regime-awareness integration:** Introduce market regime classification into the training process using methods such as Hidden Markov Models (HMMs) or clustering-based classifiers. Regime-awareness can be embedded through adjustable parameters that favour different market conditions. The optimiser can then dynamically switch between pre-trained portfolios or trigger retraining when a regime shift is detected.

- **Enhanced asset subset selection:** The current asset selection process occurs during the preliminary backtest stage, relying primarily on performance data from the most recent year to eliminate underperforming assets. During regime shifts, this approach may yield a suboptimal asset universe. To address this, extend the lookback window to incorporate longer-term data while applying higher weighting to more recent observations to preserve responsiveness to current market conditions. Regime-awareness should also be integrated into this stage.

- **Improved expected return estimation:** Replace or augment the current return estimation approach with more sophisticated techniques, such as the Black-Litterman model or machine learning based predictive models. These methods aim to provide more stable and realistic return expectations, thereby enhancing HRP allocation performance across diverse market environments.

Collectively, these enhancements should increase the stability and consistency of the optimiser, enabling the development of a production-ready trading strategy platform.

Further enhancement of the application could focus on expanding both its analytical capabilities and operational scope. Key development directions include:

- **Integration with live data and execution APIs:** Connecting to broker platforms such as Interactive Brokers or Alpaca to enable paper trading and live order execution.

- **Machine learning integration:** Incorporating predictive models for signal generation and adaptive portfolio weighting using reinforcement or deep learning approaches.
- **Advanced risk modeling:** Implementing comprehensive downside risk and stress-testing modules, including Value-at-Risk (VaR), Conditional VaR, and factor exposure analysis.
- **Enhanced user interface:** Developing interactive dashboards and real-time monitoring tools for both backtesting and live trading environments.
- **Modular plugin architecture:** Enabling users to design and integrate custom strategy scripts or optimisation routines through a standardized API.

By pursuing these extensions, the platform could evolve from a research-focused analytical framework into a full-featured algorithmic trading environment. Its modular architecture and data-driven foundation provide a solid base for continued innovation in quantitative finance.

Appendix A Application Setup and Code Layout

This section documents system requirements, installation, and the structural organization of the project.

A.1 Prerequisites

- Python 3.10+
- Node.js 22.x
- SQL Server 2022
- SQL Server Management Studio
- git

A.2 Installation Instructions

1. Clone the project repository:

```
git clone https://github.com/barneytodd/quantApp.git
cd quantApp

```

Listing 1: Clone the repository

2. Create and activate a virtual environment:

```
cd backend
python -m venv venv
source venv/bin/activate  # (Mac/Linux)
venv\Scripts\activate     # (Windows)

```

Listing 2: Create and activate virtual environment

3. Install python dependencies: In the backend folder run:

```
pip install -r requirements.txt

```

Listing 3: Install python dependencies

4. Install npm dependencies: In the frontend folder run:

```
npm install

```

Listing 4: Install dependencies

5. Set up database:

   - Open SSMS and connect to SQL Server using windows authentication
   - Create database by running:

```
CREATE DATABASE QuantDB;
GO

```

Listing 5: Create database

   - Add a new user and set a password:

```
CREATE DATABASE QuantDB;
GO

```

Listing 6: Create database

   - Add a new user In SSMS, expand Security -> Logins, Right click on Logins -> New login, Set user name, Select SQL Server Authentication and set a password, Set the Default Database to QuantDB, Click OK
   - Configure backend for the new database In database.py set:

```
SQLALCHEMY_DATABASE_URL = (
"mssql+pyodbc://{YourUserName}:{YourPassword}@localhost\\SQLEXPRESS/QuantDB?driver=ODBC+Driver+17+for+SQL+Server"
)

```

Listing 7: Configure backend for db

   In database_async.py set:

```
dsn = (
"Driver=ODBC Driver 17 for SQL Server;"
```

```
                    "Server=localhost\\SQLEXPRESS;"
                    "Database=QuantDB;"
                    "UID={YourUsername};"
                    "PWD={YourPassword};"
                    "TrustServerCertificate=yes;"
                    )

```

Listing 8: Configure backend for db

6. Run the backend: In the backend folder run:

```
uvicorn app.main:app --reload

```

Listing 9: Run the backend using uvicorn

7. Run the frontend: In the frontend folder run:

```
npm start

```

Listing 10: Run the frontend

8. Access the application locally:

```
http://localhost:3000

```

Listing 11: Access the local app

A.3 Project Structure

```
\begin{lstlisting}[caption={Backend directory structure}]
    backend/
    |
    +-- main.py               # Main entry point for the backend
    +-- api/                  # API routes and logic
    |   +-- routes/           # Route definitions (backtesting, data, portfolio)
    |
```

```
              +-- crud/                 # Database CRUD operations
              |
              +-- database/             # Database connection and async
setup
              |
              +-- models/               # ORM / data models
              |
              +-- schemas/              # Pydantic schemas for request/
response
              |
              +-- services/             # Business logic, computations,
and pipelines
              |   +-- backtesting/      # Backtesting engines, helpers,
tasks
              |   +-- data/             # Data fetching and preprocessing
              |   \-- portfolio/        # Portfolio stages and
calculations
              |
              +-- stores/               # Task or temporary storage
utilities
              |
              +-- strategies/           # Trading strategy definitions
              |
              \-- utils/                # Helper functions and utilities

```

Listing 12: Backend structure

```
              frontend/
              |
              +-- App.js, App.css          # Main React app entry point
              +-- index.js, index.css      # React DOM entry and global
styles
              +-- logo.svg                 # App logo
              |
              +-- components/              # Reusable UI components
              |   +-- layout/              # Layout components (Sidebar)
              |   \-- ui/                  # UI widgets (cards, sliders,
selects)
              |
```

```
             +-- pages/                  # Page-level components
             |   +-- Backtesting/        # Backtesting page
             |   |   +-- charts/         # Chart components and
calculations
             |   |   +-- components/     # Backtesting UI components
             |   |   +-- hooks/          # Custom React hooks for
backtesting
             |   |   +-- parameters/     # Backtesting parameters
             |   |   \-- tabs/           # Page tabs (Overview, Risk,
Trade Analytics)
             |   |
             |   +-- Dashboard/          # Dashboard page
             |   |   \-- chart/          # Dashboard charts
             |   |
             |   +-- PortfolioAnalysis/  # Portfolio analysis page
             |   |   +-- components/     # Stage components
             |   |   +-- hooks/          # Custom hooks
             |   |   \-- params/         # Page-specific parameters
             |   |
             |   \-- TradingSimulator/   # Trading simulator page
             |       \-- hooks/          # Hooks for portfolio loading
             |
             \-- utils/                  # Utility functions (
indicators, helpers)

```

Listing 13: Frontend directory structure