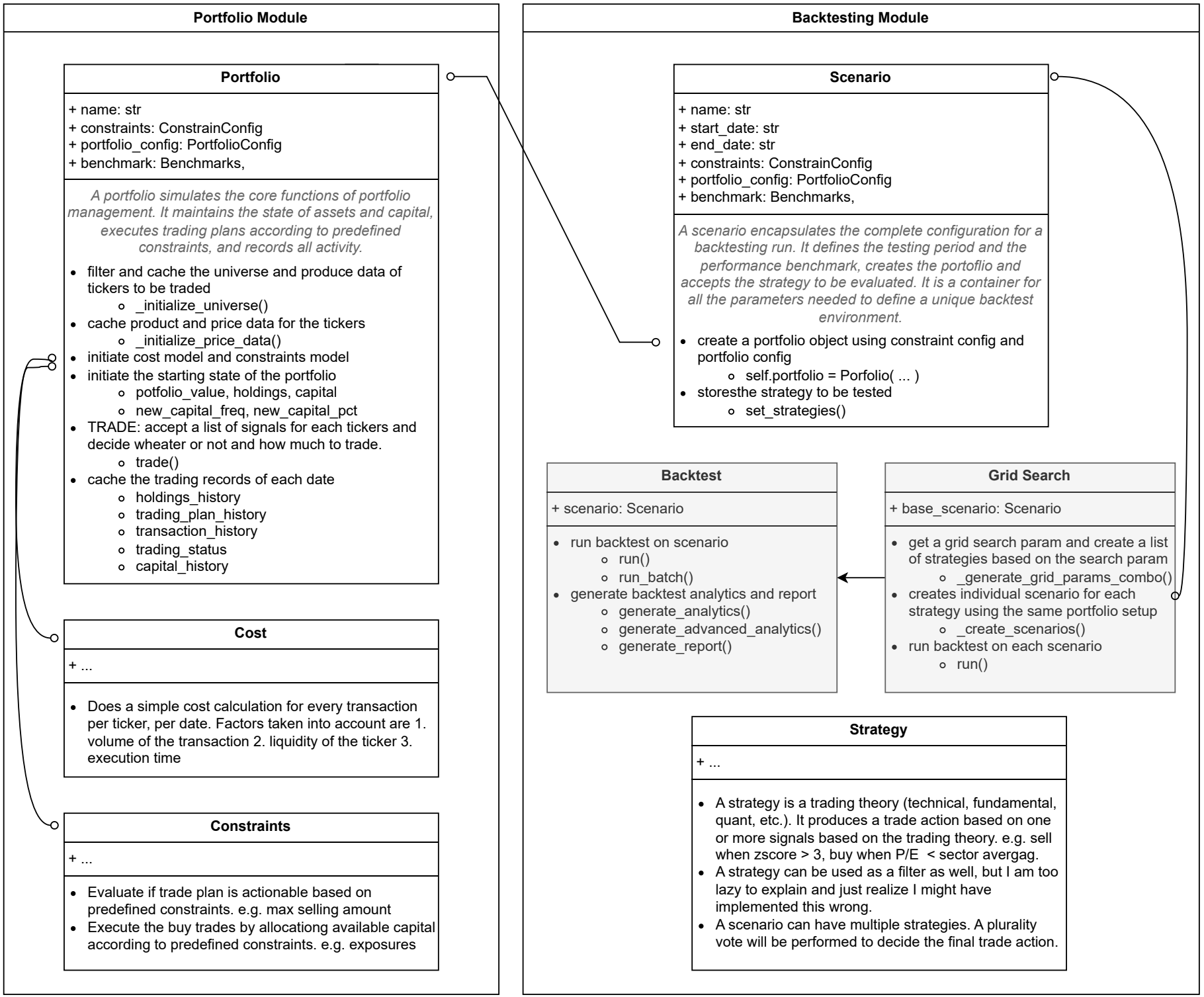
Portfolio Module
Backtesting Module
Portfolio
+ name: str
+ constraints: ConstrainConfig
+ portfolio_config: PortfolioConfig
+ benchmark: Benchmarks,
A portfolio simulates the core functions of portfolio management. It maintains the state of assets and capital, executes trading plans according to predefined constraints, and records all activity.
filter and cache the universe and produce data of tickers to be traded
_initialize_universe()
cache product and price data for the tickers
_initialize_price_data()
initiate cost model and constraints model
initiate the starting state of the portfolio
potfolio_value, holdings, capital
new_capital_freq, new_capital_pct
TRADE: accept a list of signals for each tickers and decide wheater or not and how much to trade.
trade()
cache the trading records of each date
holdings_history
trading_plan_history
transaction_history
trading_status
capital_history
Cost
+ ...
Does a simple cost calculation for every transaction per ticker, per date. Factors taken into account are 1. volume of the transaction 2. liquidity of the ticker 3. execution time
Constraints
+ ...
Evaluate if trade plan is actionable based on predefined constraints. e.g. max selling amount
Execute the buy trades by allocationg available capital according to predefined constraints. e.g. exposures
Scenario
+ name: str
+ start_date: str
+ end_date: str
+ constraints: ConstrainConfig
+ portfolio_config: PortfolioConfig
+ benchmark: Benchmarks,
A scenario encapsulates the complete configuration for a backtesting run. It defines the testing period and the performance benchmark, creates the portoflio and accepts the strategy to be evaluated. It is a container for all the parameters needed to define a unique backtest environment.
create a portfolio object using constraint config and portfolio config
self.portfolio = Porfolio(...)
storesthe strategy to be tested
set_strategies()
Backtest
+ scenario: Scenario
run backtest on scenario
run()
run_batch()
generate backtest analytics and report
generate_analytics()
generate_advanced_analytics()
generate_report()
Grid Search
+ base_scenario: Scenario
get a grid search param and create a list of strategies based on the search param
_generate_grid_params_combo()
creates individual scenario for each strategy using the same portfolio setup
_create_scenarios()
run backtest on each scenario
run()
Strategy
+ ...
A strategy is a trading theory (technical, fundamental, quant, etc.). It produces a trade action based on one or more signals based on the trading theory. e.g. sell when zscore > 3, buy when P/E < sector avergag.
A strategy can be used as a filter as well, but I am too lazy to explain and just realize I might have implemented this wrong.
A scenario can have multiple strategies. A plurality vote will be performed to decide the final trade action.