Peak_Trade – Vollständige Systemdokumentation

Peak_Trade System – Gesamtübersicht & Implementationsdokumentation

Inhalt

- Option A: Strategy-Fix (State-Signale)
- Option B: Verbesserungen & Validierung
- Option C: Dynamisches Laden der Strategien
- Option D: Filtersystem & Risk-Module
- Option E: Engine-Optimierung + BacktestResult
- Option F: Unit Tests
- Projektstruktur
- Beispielcode & Nutzung

Option A – Strategy-Fix (State-Signale)

Die `ma_crossover.py` wurde vollständig überarbeitet.

Wichtigster Bestandteil:

- Event-Signale (1/-1) → State-Signale (1/0)
- Crossover über MA-Differenz
- robuster Umgang mit NaN
- Parametervalidierung

Rückgabeformat:

`pd.Series` mit Werten `0` (flat) oder `1` (long).

Option B – Verbesserungen & Validierung

- MA-Perioden validiert (`fast < slow`)

- Datenlänge validiert (`len(df) >= slow`)

- Logging optional möglich

- Nutzen der `min_periods`-Option bei MA-Berechnung

Option C – Dynamisches Laden der Strategien

In `src/strategies/__init__.py`:

```python
STRATEGY_REGISTRY = {
    "ma_crossover": "ma_crossover",
    "momentum": "momentum",
    "rsi": "rsi",
    "bollinger": "bollinger",
    "macd": "macd",
    "ecm": "ecm",
}

def load_strategy(strategy_name: str):
    module = __import__(f"src.strategies.{STRATEGY_REGISTRY[strategy_name]}", fromlist=["generate_signals"])
    return module.generate_signals
```

Option D – Filter & Risk-System

Risk-Module (`position_sizer.py`)

- nutzt `PositionRequest`
- berechnet dynamische Positionsgrößen basierend auf:
  - Equity
  - Entry Price
  - Stop-Loss Price
  - Risk per Trade

Filter-Beispiel:

`apply_volume_filter(signals, df, threshold)`

Option E – Engine-Optimierung (BacktestEngine)

BacktestResult hinzugefügt:

- Sharpe Ratio
- Max-Drawdown
- Profit-Factor
- Win-Rate
- Equity-Curve (`pd.Series`)
- Liste der Trades

Engine-Integration:

- dynamisches Laden der Strategie
- Signale generieren
- Kauf-/Verkaufssimulation
- Equity-Tracking

- Result-Objekt erzeugen

Option F – Unit Tests

Test für Signals:

- erzeugt korrekte 0/1 State-Signale

- robust gegen Trendwechsel

Test für Engine:

- führt Backtest mit Testdaten aus

- erwartet mindestens 1 Trade

- testet Equity-Kurve

Projektstruktur

```
Peak_Trade/
■■■ src/
■  ■■■ strategies/
■  ■  ■■■ ma_crossover.py
■  ■  ■■■ __init__.py
■  ■■■ backtest/
■  ■  ■■■ engine.py
■  ■  ■■■ results.py
■  ■  ■■■ stats.py
■  ■■■ risk/
■  ■  ■■■ position_sizer.py
```

■ ■■■ data/

■■■ config.toml

■■■ tests/

■ ■■■ test_ma_crossover.py

■ ■■■ test_engine_basic.py

■■■ README.md

```

---

## Nutzung

Beispiel:

```python

from src.backtest.engine import BacktestEngine

engine = BacktestEngine()

result = engine.run_realistic(df, "ma_crossover", params)

print(result.stats)

print(result.equity_curve.tail())

```

Hinweise

Diese Dokumentation fasst alle Optimierungen zusammen und beschreibt die vollständig funktionieren
de Systemarchitektur von Peak_Trade.
```