**Key Components**

**1. Data Retrieval (Yahoo Finance API)**

* **Challenges:**
  + API restrictions (7-day batch, 30-day window).
  + Dependency on reliable API performance and internet connectivity.
* **Solution:**
  + Implemented chunking for 2-week retrieval.
  + Supports extension to daily automation in production.
* **Customization:**
  + End date adjustable; users must handle overwriting issues by defining unique paths in definitions.py.

**2. Base Strategy Interface (base\_strategy.py)**

* Acts as a blueprint for buy/sell signal generation.
* Facilitates scalability to other strategies.

**3. Momentum Strategy (momentum\_strategy.py)**

* **Core Logic:**
  + Implements a momentum indicator.
  + **Parameters:**
    - Rolling average period (Eg: 5 minutes).
    - Threshold (Eg: 0.0005).
  + Generates signals based on threshold deviations.
* **Customization & Testing:**
  + Thresholds and periods can be adjusted for tuning.

**4. Backtesting Engine (backtester.py)**

* **Inputs:**
  + Strategy class (e.g., Momentum Strategy).
  + SPX data (with Datetime and Close columns).
  + Initial capital, commission, and position-closing time.
* **Execution:**
  + Simulates trades iteratively over time.
  + Handles pending trades by rolling them to the next day.
* **Outputs:**
  + Real-time trade information.
  + Performance statistics via print\_performance().

**5. Error Handling and Reliability**

* Logging for error, debug, and info messages.
* Exception handling for missing data or edge cases.
* Unit tests ensure robustness and verify logic.

**Assumptions and Limitations**

1. **Pending Trades:**
   * Trades opened close to the market's closing time (e.g., 15:50:00–15:59:00) are carried forward and closed the next day.
   * Assumes availability of accurate opening prices for the next day.
2. **Data Accuracy:**
   * Relies on Yahoo Finance API's data integrity and update frequency.
3. **Scalability:**
   * Designed for a single strategy. Requires additional development for multi-strategy or multi-asset capabilities.

**Potential Enhancements**

1. **Automation:**
   * Automate data retrieval and backtesting runs for live trading scenarios.
   * Implement cloud storage for historical data.
2. **Parameter Optimization:**
   * Use techniques like grid search or genetic algorithms to optimize thresholds and rolling periods.
3. **Multi-Asset Support:**
   * Extend the system to include other indices or instruments.
4. **Error Handling:**
   * Improve resilience by pre-checking data gaps and handling API downtime gracefully.