# Algorithmic trading Strategy Development

**Objective:** Developing and backtesting an algorithmic trading strategy using historical price data.

**Tasks:**

- Create a momentum-based, mean-reversion, or arbitrage trading strategy.
- Use technical indicators (e.g., MovingAverages, RSI, Bollinger Bands).
- Evaluate performance using metrics like Sharpe ratio, drawdowns, and alpha.

**DataSets:**

1. [Yahoo Finance API] (https://finance.yahoo.com/) for historical stock, ETF, and index prices.
2. [Quandl] (https://www.quandl.com/) for financial data like interest rates, commodities, or cryptocurrencies.

1. **Definition of Trading Strategy**
    - **Momentum-Based Strategy:** Buy when the price is trending upward, and Sell when it is trending downward. Use indicators like Moving Average convergence Divergence (MACD) or Relative Strength Index (RSI) to identify momentum.
    - **Mean-Reversion Strategy:** Assume prices will revert to their historical mean. Use Bollinger Bands to determine overbought/oversold conditions.
    - **Arbitrage Strategy:** Identify pricing inefficiencies between correlated assets, e.g., pairs trading using cointegration tests.

2. **Fetching Historical Data**
    - From Yahoo Finance API or Quandl download historical price data. Start by pulling the data for stocks, ETFs, indices, commodities, or cryptocurrencies, depending on the strategy requirement.
    - Script to extract and preprocess the data, dealing with missing values, adjusting for splits, and normalizing the data.

3. **Technical Indicator Calculation**

    Calculate relevant indicators for the chosen strategy:

    i. **Momentum-Based**: Compute Moving Averages (SMA/EMA), RSI, MACD, etc.
    ii. **Mean-Reversion**: Compute Bollinger Bands or Z-scores.

iii. **Arbitrage**: Calculate price ratios for pairs trading and perform cointegration tests.

Use Python libraries like **pandas_ta** or **TA-Lib** for technical indicator calculations.

4. **Backtesting Framework**
   - Develop a backtesting framework to evaluate the strategy's performance.
   - Components of a backtesting framework:
     1. Buy/Sell signal generation based on indicators.
     2. Order execution logic, e.g., market orders, stop-loss, take-profit.
     3. Portfolio management, including position sizing.
   - Python libraries like **Backtrader**, **Zipline**, or **PyAlgoTrade** can help you create a robust backtesting environment.

5. **Performance Evaluation**
   - Now we calculate the performance metrics using following parameters:
     1. Share Ratio: Measure the risk-adjusted return
     2. Maximum Drawdown: Assess the largest drop in portfolio value.
     3. Alpha: Compares the strategy's performance against a benchmark index.
   - Can be done using libraries like numpy and pandas for computations and matplotlib for visualization.

6. **Optimize and Tune Parameters**
   - Using grid search or other optimization methods to tune the strategy parameters (e.g., length of moving averages).
   - Avoid overfitting by testing on out of sample data (train-test split).

7. **Report and Analysis**
   - Summarizing the results, including visualizations like equity curve, rolling sharpe ratio, drawdown chart, etc.
   - Discussion of strategy's strengths and weaknesses, and suggestions for possible improvements.