## **Group Assignment 3: Post-Trade Analysis**

The overall framework will consist of three main components:

1. **Data Pre-processing**: This module will be responsible for preparing and cleaning the data, ensuring it is in the proper format for trading simulations.
2. **Trading Execution**: A class that will instantiate and execute trading strategies based on the pre-processed data. This component should be able to handle different types of trading strategies and ensure accurate execution of trades.
3. **Post-Trade Analysis**: After trading simulation, this module will analyse the returns and performance of the strategy. It will calculate various metrics such as the Sharpe ratio, and generate visualisations to help in assessing the strategy’s effectiveness.

### **Part 1: Post-Trade Analysis Class**

This class will handle various post-trade analysis tasks. Here’s a breakdown of the methods and functionalities it should include:

1. **Initialization**:
   * **Input**: A Pandas Series or DataFrame containing return series.
   * **Output**: Instantiation of the analysis class.
2. **Cumulative Returns**:
   * **Function**: Calculate and return the cumulative returns of the trading strategy.
3. **Maximum Drawdown**:
   * **Function**: Calculate and return the maximum drawdown experienced by the strategy.
4. **Sharpe and Sortino Ratios**:
   * **Function**: Calculate and return the Sharpe and Sortino ratios to assess the risk-adjusted returns.
5. **Hit Ratio**:
   * **Function**: Calculate the percentage of profitable trades.
6. **Monthly Returns Heatmap**:
   * **Function**: Generate a heatmap of monthly returns for each year to visualise performance variability over time.
7. **Plots**:
   * **Function**: Generate plots for cumulative returns and other relevant financial metrics.

### **Part 2: Web Interface Development**

Integration of the post-trade analysis into the web interface should include the following features:

1. **New Web Page for Post-Trade Analysis**:
   * **Page Content**: This page should display all calculated metrics in a table format and provide visual plots (like heatmaps and cumulative returns graph).
   * **Functionality**: A button labelled "Run Post-Trade Analysis" which, upon clicking, instantiates the Post\_Trade\_Analysis class with the return series generated by the trading strategy executed in the previous step.
2. **User Interaction**:
   * **Inputs**: None needed explicitly for this page as it uses the results from the executed trade.
   * **Outputs**: Display the post-trade metrics and visualisations as described.

### **Submission Requirements**

1. **Code**: You will need to submit a Python file named post\_trade\_analysis.py containing the class and functions detailed above. This should be integrated with the existing web interface files.
2. **Web Interface**: Ensure that all files necessary to run the updated web interface are included. This should be user-friendly and allow for easy navigation between the pre-processing, trading execution, and post-trade analysis pages.
3. **Demonstration Video**: Create a concise video demonstrating the entire workflow: from data input and preprocessing, through trading execution, to post-trade analysis, with 1 strategy of your choice.

Other Things that can be done -

Implement functionality to split the historical data into training and testing sets, then use different stocks to evaluate strategy on and report comparison on which stock it performed well.

Utilize the talib library to compute the RSI and EMA indicators. Develop a strategy that combines these indicators to generate buy and sell signals. For example, a buy signal could be triggered when the RSI is below 30 (indicating oversold conditions) and the price is above a certain EMA, suggesting an upward trend.

Incorporate the trading period visualisation in post trade class where the goal is to have time series plotted with an up and down arrow where the signal was to enter and exit with green and red colours.