Design and implement a low latency high frequency trading system. Candidates need to demonstrate their understanding of the trading system and related infrastructure. Saccade will provide sample market data for the task.

## **Technical requirements**

* **Linux**
  + If you want to use Mac or WSL, make sure it's working in Linux before submission
* **Modern C++ in GCC or Clang**
* **Standard library, no third party libraries, i.e. you cannot use boost libraries**
* Version control the source code with local git and submit the local git repo for final review (do not upload on the public cloud).
* To submit, please compress all source codes and documents and attach the compressed file to an **email** and reply to all cc-ed people. Please don’t include market data or compiled binary/firmware

## **Functional requirements**

### **Mandatory**

* Two market data injectors which read market data from sample files and send it through the network as multicast.
  + each injector runs as a separate process
* Two mock exchanges which accept orders and send back responses through TCP.
  + Each mock exchange runs as a separate process
  + Mock exchanges don't need to implement matching logic. It can simply reply with order fill every time.
* One strategy module which supports multiple channel for real-time market data, order execution
  + The strategy module should be able to receive multicast data from two market data injectors and send orders to two mock exchanges through TCP
  + Trading logic is not the scope of the test evaluation, any simple logic that can prove the platform's functionality is fine
  + Support order life cycle management in order execution
* Reasonable level of modularization

### **Strongly recommended**

### Support order book building with order-based market data in strategy module

* Minimize tick-to-trade latency and provide the way to measure

### **Optional**

* Supports pre-order risk limit checking and monitoring
* Efficient record keeping of order/trade/market data

## **Input Data**

* 2 files for two different symbols
* File format - epoch, order id, symbol, order side, order category, price, quantity
* order id is unique per order book per side.

## **Expected Result**

* Source codes
  + Implementation for mandatory requirements
  + Implementation or Interface/header file for optional requirements
* Design document
  + Why do you make a certain design choices and what was the alternative
  + What has been implemented and detailed plans of what you would have implemented if you have more time
* Compile and execution instruction

## **Evaluation Criteria**

* General level of effort
  + Fulfilled functional requirements
  + Testing, robustness of the implementation, commenting
* Software design and documentation
  + Design principles and efficiency of the implementation
  + Readability/structure/modularization of the code
  + Quality of the description of approach in documentation
* Complexity/Domain knowledge
  + Understanding of network programming
  + Understanding and handling of market data and order lifecycle
  + Proper and efficient usage of C++
  + Understanding of low latency system development best practices