Position Monitoring System Guide

Author : Tommy Lee
Date : 29-Jun-2023

Thank you for taking the time to review my solution. There are two parts in this guide. Part 1 is the developer's guide. Part 2 is the user's guide. If you have any questions please feel free to contact me.

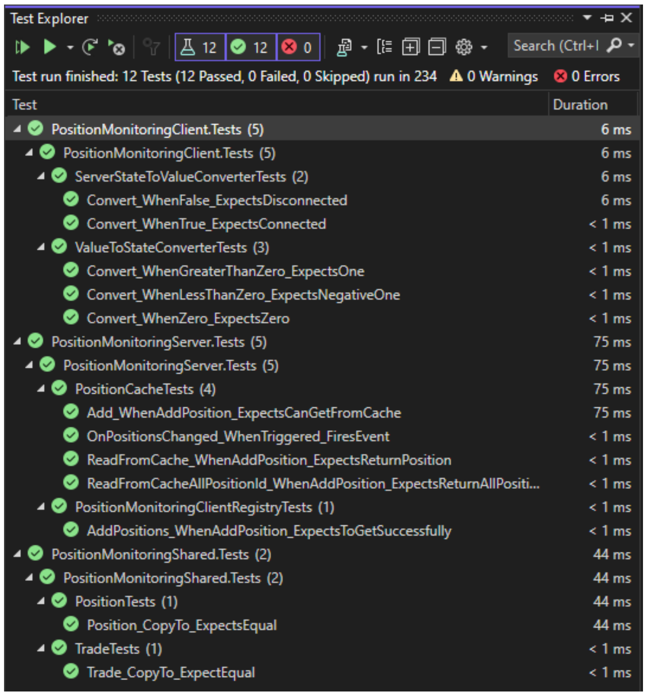
Part 1 - Developer's Guide

- The source code is in PositionMonitoringSolution.zip. It is developed using Visual Studio 2022 and .NET 6.
- Unzip this and open PositionMonitoring.sln in Visual Studio 2022.
- There are six projects :
    - PositionMonitoringClient - This contains the WPF UI to display the real-time positions and trades data.
    - PositionMonitoringClient.Tests - This contains the nunit tests for project PositionMonitoringClient
    - PositionMonitoringServer - This contains the Services to publish the real-time positions and trades data to the clients.
    - PositionMonitoringServer.Tests - This contains the nunit tests for project PositionMonitoringServer
    - PositionMonitoringShared - This contains the data model and classes shared by the PositionMonitoringClient and PositionMonitoringServer projects.
    - PositionMonitoringShared.Tests - This contains the nunit tests for project PositionMonitoringShared
- It uses the following Nuget packages :
    - Google.Protobuf - Version="3.23.3"
    - Grpc.Core - Version="2.46.6"
    - Microsoft.NET.Test.Sdk - Version="17.6.3"
    - NUnit - Version="3.13.3"
    - NUnit3TestAdapter - Version="4.5.0"
    - Google.Protobuf Version="3.23.3"
- Please refresh the Nuget packages, and recompile all projects. If successful, you should see the following exe generated :
    - PositionMonitoringSolution\PositionMonitoringClient\bin\Debug\net6.0-windows\PositionMonitoringClient.exe
    - PositionMonitoringSolution\PositionMonitoringServer\bin\Debug\net6.0\PositionMonitoringServer.exe

- Here is additional details on the projects :
    - PositionMonitoringShared

        - Position.cs and Trade.cs - These are the data model
        - PositionMonitoring.cs and PositionMonitoringGrpc.cs - These are the auto generated classes that have been modified to consume the gRPC API specific to this project
    - PositionMonitoringServer
        - PositionCache.cs - This cache stores all the positions and trades
        - PositionMonitoringClientRegistry.cs - This registry keeps track of all the client connections
        - RealTimeMarketDataServiceSimulator.cs - This simulator generates the randomized positions and trades update
        - Program.cs - This initializes the PositionCache and RealTimeMarketDataServiceSimulator, and start publishing real-time data to clients.
    - PositionMonitoringClient
        - MainViewModel.cs - This is the viewmodel of the view. It establish a connection to PositionMonitoringServer, and process the real-time updates from PositionMonitoringServer.
        - MainWindow.xaml - This is the view. It contains a Positions table and a Trades table to display the real-time updates.
        - PositionViewMode.cs and TradeViewModel.cs - These are the viewmodel for the Position and Trade data model, and are used to bind to the tables in the view.
        - ServerStateToValueConverter.cs and ValueToStateConverter.cs - These are the converters used for data conversion in the view.

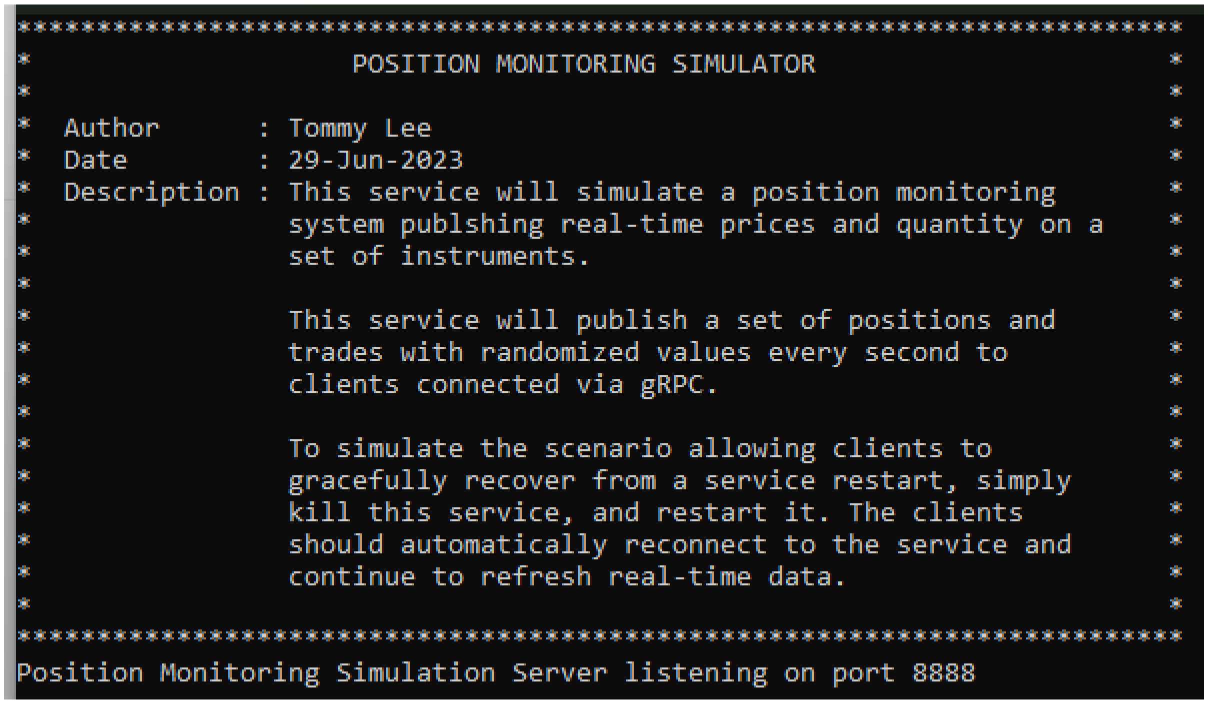
- Unit tests has been implemented for some of the classes and they should all pass



```
Test Explorer
12   12   0   Search (Ctrl+I
Test run finished: 12 Tests (12 Passed, 0 Failed, 0 Skipped) run in 234   0 Warnings   0 Errors
Test                                                                         Duration
PositionMonitoringClient.Tests (5)                                               6 ms
  PositionMonitoringClient.Tests (5)                                             6 ms
    ServerStateToValueConverterTests (2)                                         6 ms
      Convert_WhenFalse_ExpectsDisconnected                                      6 ms
      Convert_WhenTrue_ExpectsConnected                                        < 1 ms
    ValueToStateConverterTests (3)                                             < 1 ms
      Convert_WhenGreaterThanZero_ExpectsOne                                   < 1 ms
      Convert_WhenLessThanZero_ExpectsNegativeOne                              < 1 ms
      Convert_WhenZero_ExpectsZero                                             < 1 ms
PositionMonitoringServer.Tests (5)                                              75 ms
  PositionMonitoringServer.Tests (5)                                            75 ms
    PositionCacheTests (4)                                                      75 ms
      Add_WhenAddPosition_ExpectsCanGetFromCache                                75 ms
      OnPositionsChanged_WhenTriggered_FiresEvent                              < 1 ms
      ReadFromCache_WhenAddPosition_ExpectsReturnPosition                      < 1 ms
      ReadFromCacheAllPositionId_WhenAddPosition_ExpectsReturnAllPositi...     < 1 ms
    PositionMonitoringClientRegistryTests (1)                                  < 1 ms
      AddPositions_WhenAddPosition_ExpectsToGetSuccessfully                    < 1 ms
PositionMonitoringShared.Tests (2)                                              44 ms
  PositionMonitoringShared.Tests (2)                                            44 ms
    PositionTests (1)                                                           44 ms
      Position_CopyTo_ExpectsEqual                                              44 ms
    TradeTests (1)                                                             < 1 ms
      Trade_CopyTo_ExpectEqual                                                 < 1 ms
```

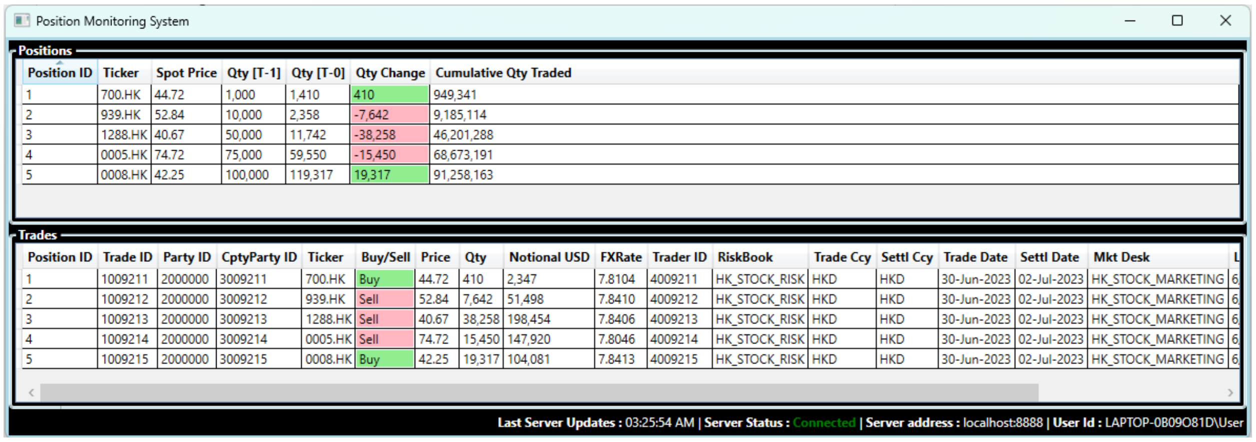
Part 2 - User’s Guide

- Start the PositionMonitoringServer by running PositionMonitoringSolution\PositionMonitoringServer\bin\Debug\net6.0\PositionMonitoringServer.exe



```
*******************************************************************
*                   POSITION MONITORING SIMULATOR                 *
*                                                                 *
*  Author      : Tommy Lee                                        *
*  Date        : 29-Jun-2023                                      *
*  Description : This service will simulate a position monitoring*
*                system publshing real-time prices and quantity on a *
*                set of instruments.                              *
*                                                                 *
*                This service will publish a set of positions and *
*                trades with randomized values every second to    *
*                clients connected via gRPC.                      *
*                                                                 *
*                To simulate the scenario allowing clients to     *
*                gracefully recover from a service restart, simply*
*                kill this service, and restart it. The clients   *
*                should automatically reconnect to the service and*
*                continue to refresh real-time data.              *
*                                                                 *
*******************************************************************
Position Monitoring Simulation Server listening on port 8888
```

- Start multiple PositionMonitoringClient by running multiple PositionMonitoringSolution\PositionMonitoringClient\bin\Debug\net6.0-windows\PositionMonitoringClient.exe
- You should see the Positions table and Trades table updating in real-time. You should also see at the bottom status bar that shows “Last Server Updates” timestamp is updating, and “Server Status” shows “Connected”.



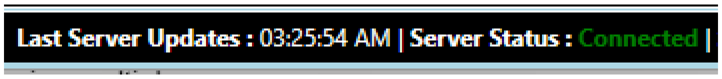
Positions

Position ID	Ticker	Spot Price	Qty [T-1]	Qty [T-0]	Qty Change	Cumulative Qty Traded
1	700.HK	44.72	1,000	1,410	410	949,341
2	939.HK	52.84	10,000	2,358	-7,642	9,185,114
3	1288.HK	40.67	50,000	11,742	-38,258	46,201,288
4	0005.HK	74.72	75,000	59,550	-15,450	68,673,191
5	0008.HK	42.25	100,000	119,317	19,317	91,258,163

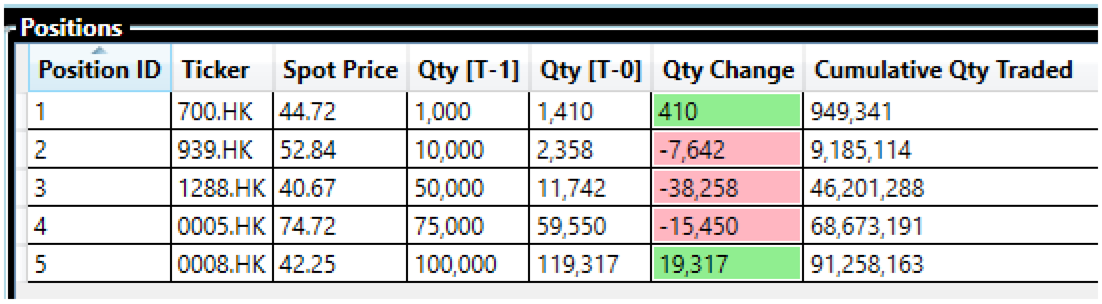
Trades

Position ID	Trade ID	Party ID	CptyParty ID	Ticker	Buy/Sell	Price	Qty	Notional USD	FXRate	Trader ID	RiskBook	Trade Ccy	Settl Ccy	Trade Date	Settl Date	Mkt Desk	L
1	1009211	2000000	3009211	700.HK	Buy	44.72	410	2,347	7.8104	4009211	HK_STOCK_RISK	HKD	HKD	30-Jun-2023	02-Jul-2023	HK_STOCK_MARKETING	6
2	1009212	2000000	3009212	939.HK	Sell	52.84	7,642	51,498	7.8410	4009212	HK_STOCK_RISK	HKD	HKD	30-Jun-2023	02-Jul-2023	HK_STOCK_MARKETING	6
3	1009213	2000000	3009213	1288.HK	Sell	40.67	38,258	198,454	7.8406	4009213	HK_STOCK_RISK	HKD	HKD	30-Jun-2023	02-Jul-2023	HK_STOCK_MARKETING	6
4	1009214	2000000	3009214	0005.HK	Sell	74.72	15,450	147,920	7.8046	4009214	HK_STOCK_RISK	HKD	HKD	30-Jun-2023	02-Jul-2023	HK_STOCK_MARKETING	6
5	1009215	2000000	3009215	0008.HK	Buy	42.25	19,317	104,081	7.8413	4009215	HK_STOCK_RISK	HKD	HKD	30-Jun-2023	02-Jul-2023	HK_STOCK_MARKETING	6

Last Server Updates : 03:25:54 AM | **Server Status :** Connected | **Server address :** localhost:8888 | **User Id :** LAPTOP-0B09O81D\User

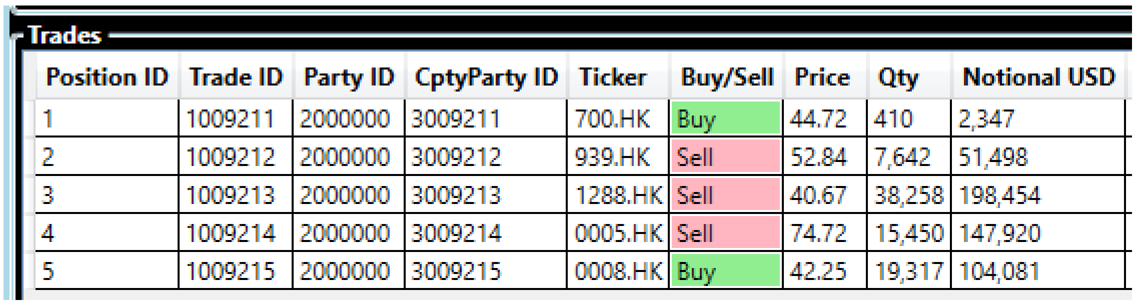

Last Server Updates : 03:25:54 AM | Server Status : Connected |

- The required functionality and optional functionality in the assignment have been implemented
- The Positions table displays the current positions with real-time updates from the server.


Positions

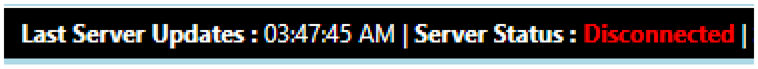
Position ID	Ticker	Spot Price	Qty [T-1]	Qty [T-0]	Qty Change	Cumulative Qty Traded
1	700.HK	44.72	1,000	1,410	410	949,341
2	939.HK	52.84	10,000	2,358	-7,642	9,185,114
3	1288.HK	40.67	50,000	11,742	-38,258	46,201,288
4	0005.HK	74.72	75,000	59,550	-15,450	68,673,191
5	0008.HK	42.25	100,000	119,317	19,317	91,258,163

- The Trades table displays the latest trade linked to the position updates. It is also displaying real-time updates from the server.

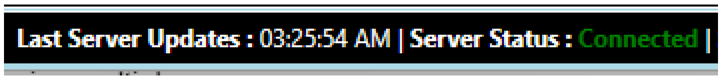

Trades

Position ID	Trade ID	Party ID	CptyParty ID	Ticker	Buy/Sell	Price	Qty	Notional USD
1	1009211	2000000	3009211	700.HK	Buy	44.72	410	2,347
2	1009212	2000000	3009212	939.HK	Sell	52.84	7,642	51,498
3	1009213	2000000	3009213	1288.HK	Sell	40.67	38,258	198,454
4	1009214	2000000	3009214	0005.HK	Sell	74.72	15,450	147,920
5	1009215	2000000	3009215	0008.HK	Buy	42.25	19,317	104,081

- To test clients gracefully recover from a server restart, stop the PositionMonitoringServer by closing it. The clients tables will stop updating, the “Last Server Updates” timestamp will stop updating, and the “Server Status” will display “Disconnected”.


Last Server Updates : 03:47:45 AM | Server Status : Disconnected |

Then restart PositionMonitoringServer, and the clients tables should immediately update with real-time data again, with the “Last Server Updates” timestamp updating, and “Server Status” displaying “Connected”.


Last Server Updates : 03:25:54 AM | Server Status : Connected |

- Several types of stress testing has been performed

- 10 clients connecting to a server for several hours and verified it is running properly
- 10 clients connecting to a server, with the server being explicitly shutdown and started multiple times, and verified it is running properly
- 10 clients connecting to a server, with the server changed to update every 100 milliseconds, and verified it is running properly

Future improvements :

- These are functionalities that I wish to complete but did not have sufficient time
  - Currently there are unit tests coverage on some classes, but the test coverage can certainly be improved
  - Currently the architecture design only supports a single instance of server, but it can be improved to support multiple instances of servers across multiple machines for load balancing and automatic failover
  - Currently the UI is in WPF, but it can be improved to also include a Web UI for supporting on different OS and devices.