ow Level Design

Our project consists of 3 Different layers: Algo Trading, Market Client and the Presentation layer.  
  
Algo Trading:  
Data [Folder]: This folder is our data layer. In this folder we created all the necessary items requests that are being sent to the server with the right variable Jason names:  
 - BuyRequest: string:type, int: price,amount,commodity

* Cancel Request: string: type, int: id
* QueryRequest: string: type, int:id
* QueryMarket: string:type, int:commodity
* QueryUser: string:type
* SellRequest: string:type, int: price,amount,commodity
* QueryMarketRequest: string: type
* queryUserRequests: string: type

Also, in the same folder we created classes for some of the server responds:

* MarketCommodityOffer : int: ask,bid
* MarketItemQuery : int:price,amount, string:type,user,commodity
* MarketUserData : float: founds, List<int>: requests, Dictionary: commodities
* MarketAll: List<MarketUnit> list
* MarketUnit: Dictionary<string,info>: info, int: id
* QueryUserRequestsRequest List<QueryUserUnit> list  
  -QueryUserUnit: Dictionary<string,Object>: request , int: id

Data-LOG [SubFolder]:

In this folder there are 2 files connected to the operation of log4net:

* LogHelper: contains three functions that returns the "logger" property of the log4net expension.
* [callerFilePath]string filename ="“: returns the location of the file as the logger.
* string Data – returns a string as the logger according to the programmer choose.
* informative logger: returns a combination of the 2 (string with location as the logger).
* LoopCounter: In order to number each record created with the log4net we created a counter, that we defined as a new property for our log.

Communication [Folder]:

MarketClientOptions:

This class implements "ImarketClient" interface in MarketClient.

Each method creates a connection with the market server getting a respond from it.   
One for each of the market possible functions, while using the classes we described above.  
  
We used the implemented "SendPostRequest" method that was created by the project instructors to make the actual communication between the user and the server in each method.

Logic [Folder]:

This folder contains the AI algorithm. The algorithm connects to the sql server and retrieve the average price of commodities 4, 5,6 . Then it make buy request for them with a price lower than the average price.  
after that it sells them for higher price than the average.

We limited the investment of each commodity to 1/9 from our founds.

And when the profit hit 600 the investment in each commodity is increased by 200.

NunitTest [Folder]:

We had built tests for the data layer functions that checks the program communicate functions with the sever on a variety of inputs.

We also have built an ama test that check we make a his profit in minute of working.

MarketClient:

This folder is our business layer. This layer contains the code provided by the instructors for our project, and it contains the "SimpleHTTPClient" which responsible for creating the contact with the server and all the "Jason" code.

ImarketClient class: it's the interface we implemented in Algo Trading that contains all the requests functions.

To the DataEntries[Folder] we added the following files:

* IQueryUserUnit
* IQueryUserRequestsRequest
* ImarketUnit
* ImarketUserData

These are just interfaces we implemented in the Algo-Trading Data[Folder].

New Authentication:

In order to prevent "Replay Attacks" we added a new and secured authentication system. In every request being made to the server we added a "nonce" field which is a number that randomly being picked, and after being picked can't return again.  
That way we make sure each request to the server will be unique

In order to decrypt the respond from the server we used the decrypt algorithm provided in the milestone.

PresentationLayer

The gui based on wpf window and pages.

This folder is our presentation layer. We have a main window xaml which is called MainWindow, it's constructor calls the page MainMenu, which have buttons to other pages like requests, history, credits and AI.

AI page

This function activate the AI trading algorithm.

Requests

This page shows a menu of buttons of data layer available functions. When the user choose a function the program open a page that can read the user details about the actions.

History

This page show the history of the user buy and sell actions, by reading it from file.

Statistics:

This page shows graphs representing interesting information about the market.  
By using the sql server provided by the instructors.

In the code behind this page there's a function that receive a string representing the query that we want to send to the server.  
The function is used to find out the average price of each commodity,  
all the money transfer in the past week, number of purchases compared to number of sells the user make and information about the max and minimum sell price in the market.

Every graph is bind to unique button. Pressing on each button will make the graph connected to it appear on the screen.

Credits

This page show the owners names.

Log

We use log4net to follow the program exception throwing, and user interaction with it.

Open sources:

Code project, WPF Toolkit Charting Controls.

<https://www.codeproject.com/Articles/196502/WPF-Toolkit-Charting-Controls-Line-Bar-Area-Pie-Co>

Source forge, itextsharp.

<https://sourceforge.net/projects/itextsharp/>

