LLD

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 sends buy request to commodities 0-8, and sends buy request to commodities 0-8 with higher price. According to our tests it works perfectly. The algorithm can be stop after the user press the stop button. after the user press the button, an Boolean is changed and after the algorithm end the iteration, he is not starts a new one.

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].

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.

Credits

This page show the owners names.

Log

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

