Algo-trading market client HLD

3 Tier Design

The application will follow the 3-tier design structure:

* Presentation Layer
* Business Layer
* Data Layer

Each layer will have its own namespace or project.

Presentation Layer

There are two options for UI:  
**# GUI  
# CLI**

**GUI**

The GUI allows the user to have an easy and intuitive interface.   
The design focuses on simplicity and "**3 clicks principle**" (the user wouldn't need to click on more than 3 button in order to perform an action).  
  
The main window contains buttons other forms or display a message to the user, for each action.   
Buttons in the main window are divided into 4 categories for quick distinction and organization:  
Market actions

Here the user will be able to send buy/sell requests and cancel existing requests.

Information

Here the user will be able to send queries to the server about the user’s own assets, ask and bid prices of commodities and status of existing requests.

Default AMA configuration

Here the user will be able to activate and deactivate the Autonomous Market Agent.

Activation will disable the rest of the GUI to avoid synchronization issues. Deactivation will re-enable the GUI.

User AMA configuration

Here the user will be able to configure their own Autonomous Market Agent based on new rules and conditions.

Activation will disable the rest of the GUI to avoid synchronization issues. Deactivation will re-enable the GUI.

**CLI (Legacy interface)**

The CLI is the command line interface. This is a legacy build from previous milestones. It does not contain the latest features and serves as an alternative in case of issues with the GUI during development.

Business Layer

**Server Communications**

All the data is stored and processed in the Market Server. Control over the user's assets is done by sending requests for market actions to the Market Server. The business layers receives requests from the user via the GUI and returns the responses received by the internal process of the server.

Any requests sent and responses received will be logged in a history file. The **history** file will not make a distinction between **user** and **AMA** and is included in the **Data Layer**.

Encryption and Security

In order to prevent replay attacks the server supports a new authentication system that incorporates NONCE inside the encrypted message.

This new form of authentication is used by default. Usage of the old authentication method is still possible.

**Autonomous Market Agent**

The market client will hold an AMAthat is designed to intelligently operate without user input. The agent queries data on the market (either by using sending query requests or querying the SQL database), evaluates it and acts accordingly with the goal of making profit.

The user is able to configure a separate AMA

The AMA will not operate parallel to the user. Activation of the AMA will block the user from sending requests.

### **Request** **Frequency**

Due to server restrictions - the AMA will send at most 20 requests every 10 seconds.

Data Layer

**History**

User history:

All the user requests sent and their responses are recorded in a history file.

The history is viewable inside the GUI.

General history:

All market transactions that occur is saved in history Data Base.

The AMA use this information and make buy and sell request, in smarter way.

The user can access this information by statistic. It is displayed as a graphs.

**Logs**

The program has a log that documents all actions performed.

Open Source

The client uses **iTextSharp** for export of report in PDF.

The client uses **FluentScheduler** to schedule run of the AMA.

<https://github.com/fluentscheduler/FluentScheduler>