**Market Emulator – Client Application – HLD for Milestone 3**

**Introduction:**

The project supposed to serve users who wants to perform trading via network, like selling stocks of commodities and buying stocks of commodities. The application adjusted for the frequent changes in the market and enables the user to know about the changes. The application supports manual marketing (when the user performed deals) , and automatically marketing, via the AMA (auto market agent) , that performs deals with an algorithm that considering the status of the market (bid and asked price for each commodity) , the user's funds and the number of each commodity. The marketing is done by communication with sever that gets valid requests, and performed the wanted commands.

**Project's Structure:**

The project is built by the principe of 3-tier architecture. The main idea of this principle is making a modular software by dividing the main software to 3 sub-projects which everyone has a different role.

**Presentation Layer:**

The highest level, Presentation Layer (PL) responsible for communication with the user, showing outputs of user's request, and understanding his commands. If the program receives invalid command, an error message is printed.

**Business Layer:**

The second level, Business Layer (BL), responsible for getting the input, checking if the parameters of the command are valid (For example- if a user wants to buy X units of commodity Y, the layer checks if X > 0). If the input is valid, the BL responsible for sending the input to the suit method in the lower level, else- an error message is printed.

**Data Access Layer:**

The lowest level in the project, Data Access Layer (DAL), responsible for designing the request's details as a request object, which designed according to the request's kind (There is an object for every request's kind, which saved in DAL). After designing the request's details, the request is sent to the server, and the input from the server is returned to the user. Another responsibility of DAL is to save the history of user's request in a data structure and update the status of each request, if necessary (For example, accept of buy request).

The DAL is also responsible for communication with history DB, extracting information from the DB, analyze it and sending it to classes who needs this information for their goals.

In addition to those 4 sub-projects there are another 3 projects:

Log Project- responsible for documenting all sell and buy requests to a text file, that appears in LogFiles folder, in the folder of Log project. The log also responsible for documenting error messages (if there are). This file supposed to inform the developer about crises, and errors for improving the program and fixing bugs.

Auto Market Agent (AMA)- this project is responsible for running the algorithm for automatic marketing. The algorithm is based of knowledge of some parameters like bid and asked prices of all commodities and using SQL queries that get information from "history" server. Those parameters help to the auto market agent to decide if there is a reason to trade, which method to start (buy or sell), and in how many units of each commodity to trade – for making the maximum profit as possible.

GUI- the GUI is graphical interface. This part supports the same actions like in the CMD, but in more comfortable way, like graphs and charts. The user has the choice how to use the application- via the CMD or via the GUI.

Test- Test Project supposed to test chosen methods which are the most important to operate our project. The project contains class that uses "NUnit" extension of Visual Studio, that help us to write tests and to understand what methods should we improve or fix.

**Communication Mechanism**

Our project uses from the first milestone Server's services to get information, or performing requests like selling, buying etc.

Until the second milestone, when we perform a request to the server, we delivered him an object with request's details, and authentication information- username, and token that was constant.

Now, in the third milestone, we changed the communication mechanism to be safer and less breakable, by adding another authentication data- the nonce. Nonce is a random number that attached to requests. Nonce should be unique so user can not send request more than one time with the same nonce. In addition, the information that sent to the server is encrypted before sending and decrypted by the user after getting response from the server. The encryption insures us that no one else can view our requests or intercept them.

**LiveCharts**

LiveCharts (LVC) is an open-source project hosted at <https://lvcharts.net/> .

Source code on GitHub: <https://github.com/beto-rodriguez/Live-Charts> .

The project is a simple, flexible, interactive and powerful data visualization tool for the .Net Framework. We used LVC to visualize information for the user, by creating colorful and easy-to-understand charts and graphs.