# Trading Software Assignment

This software was coded with [Poco Framework](http://pocoproject.org/) to speed up development.

I chose this framework because while I wanted a small and lean REST framework, I also wanted some helper libraries to manipulate JSON and other amenities, while working in a familiar environment for me (Ubuntu / MySQL). Having made that choice, I bumped into several limitations and bugs in the framework at the start of development, until I realized that Ubuntu has a very old version of the framework available.

Also, I had some problems looking for documentation and samples for some of the API, having to resource to read the framework’s source code many times.

Having experienced some of these problems, I chose to make the code very minimalistic in order to reach the deadline provided while not sacrificing code quality and unit testing. I also decided to skip using this framework for database connectivity, in order to avoid unexpected issues, resorting to MySQL’s [C++ Connector](http://dev.mysql.com/doc/connector-cpp/en/connector-cpp-introduction.html).

I chose **not** to use [Wt](http://www.webtoolkit.eu/wt) as suggested in the challenge, because the API seemed too complicated for such a small project, and having no experience on this framework it would delay my progress a lot. I opted to resort to JQuery and very little amount of Javascript code for the UI. Javascript is very commonplace nowadays thanks to many available browsers in almost any popular platform, so dealing with UI components on a low level language like C++ seemed too overkill for the task at hand.

**Code Layout**

The directory of the application is composed of the following folders:

trading/ ->

database/ // Database schema and stock data

static/ // For content to be served to client browsers as it is (js libraries, etc)

templates/ // For HTML content served by http handlers

trading/ ->

src/ // Source code for the application

testsuite/ // Source for unit tests

**Design**

Basically the application is a standalone server application (no apache or nginx requirement), listening to the port 9980 (ServerApp::main()).

Incoming requests are received and passed to the HandlerFactory class, which routes them to specific handlers in the namespace trading::handlers. All the REST services required by the specification are placed there.

These services are in charge of database access, there are no other places accessing the database except to authenticate users.

This application is mostly dedicated to serve REST requests, only very minimal code handles client UI requirements. The idea behind this is that UI code can be transformed to any other framework available with very minimal effort. Also we can choose to use nginx to serve static files and act as a proxy if the need arises.

**Security**

This application supports a very basic kind of authentication service. This is **not** a secure service by any means, since it serves unencrypted content, and passwords are passed back and forth in unsecure channels. This is totally unacceptable for a production environment, and was left as is only because of time constraints and due to security not being the focus of this challenge. This being said, user provided input is never passed as-is to SQL as we use prepared statements which avoid SQL injection attacks, and a basic authorization framework verifies the user credentials on each REST request that needs to be authenticated.

**TODO**

This is the list of things left out to implement because of lack of time:

* Validate user provided input on the UI and backend.
  + very little (mostly unexistant) input validation to make sure that input is in ranges allowed was done.
* Update a trader’s cash balance on buy/sell operations
* More unit tests for edge cases on all the operations.
* Measure application performance
  + the was no focus on premature optimization at all
* Improve security and implement a more user friendly authentication service
  + Logging out is not supported, the browser needs to be closed to login and register with other credentials
* Use some kind of ORM
  + this kind of application would benefit a lot design-wise with an ORM framework
* Use better UI widgets to improve usability.
* Better error reporting
  + the application currently has a very weak error reporting system which doesn’t help tracking of errors.
* Functionality speaking the application is just an interpretation on what a trading software involves. Having no experience in this kind of software whatsoever, it’s surely in need of future functional improvement.