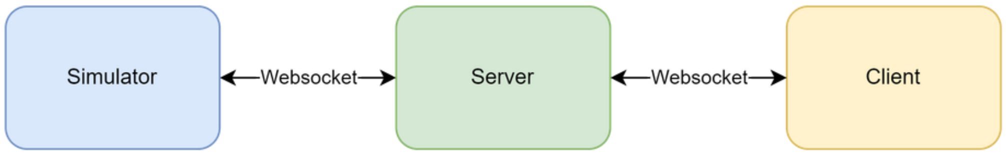
# Project 4 - Real Time API

This project requires you to create a application that publishes real time data via an API.

To do this you will first simulate real time data which you feed to a server, and they display synthesised data from the real-time data.

The data you generate will intraday data with price and quantity information



## Simulator

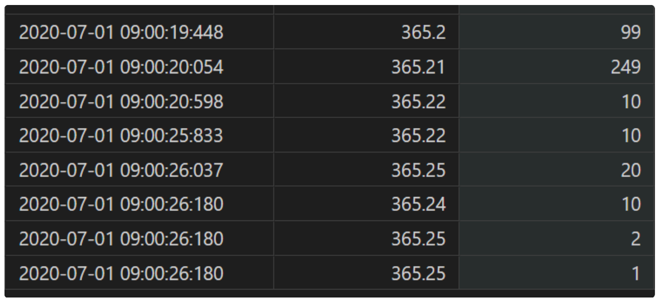
Simulators are very valuable to be able to test software and to back test algorithms.

The simulator should read a file with trade data and then replay those trades in real-time (or at a configurable rate). Some example stock trade files are provided that can be used to replay a day's trading.

The simulator should provide a websockets interface to push the data in real-time.

Since websockets have a limit to the size of data that they pass, you should also provide an HTTP interface to get trades already created at the time connection, and then send updates via websockets (no data should be lost).

Below is an example of trades in the provided trade file (time, price, quantity):



| | | |
|---|---|---|
| 2020-07-01 09:00:19:448 | 365.2 | 99 |
| 2020-07-01 09:00:20:054 | 365.21 | 249 |
| 2020-07-01 09:00:20:598 | 365.22 | 10 |
| 2020-07-01 09:00:25:833 | 365.22 | 10 |
| 2020-07-01 09:00:26:037 | 365.25 | 20 |
| 2020-07-01 09:00:26:180 | 365.24 | 10 |
| 2020-07-01 09:00:26:180 | 365.25 | 2 |
| 2020-07-01 09:00:26:180 | 365.25 | 1 |

The simulator should create an update every second, and send all the trades executed within that second.

## Server

The server should connect to the simulator and consume all trades from the simulator.

The server should aggregate data by minute, providing max and min price in each minute, and the volume traded (additional data for teams of 3, see below).

The server should accept connections from one or more browsers and push data to each browser in real-time. When a connection is lost, data should no longer be sent to that connection.

The published data should update every second.

## Client

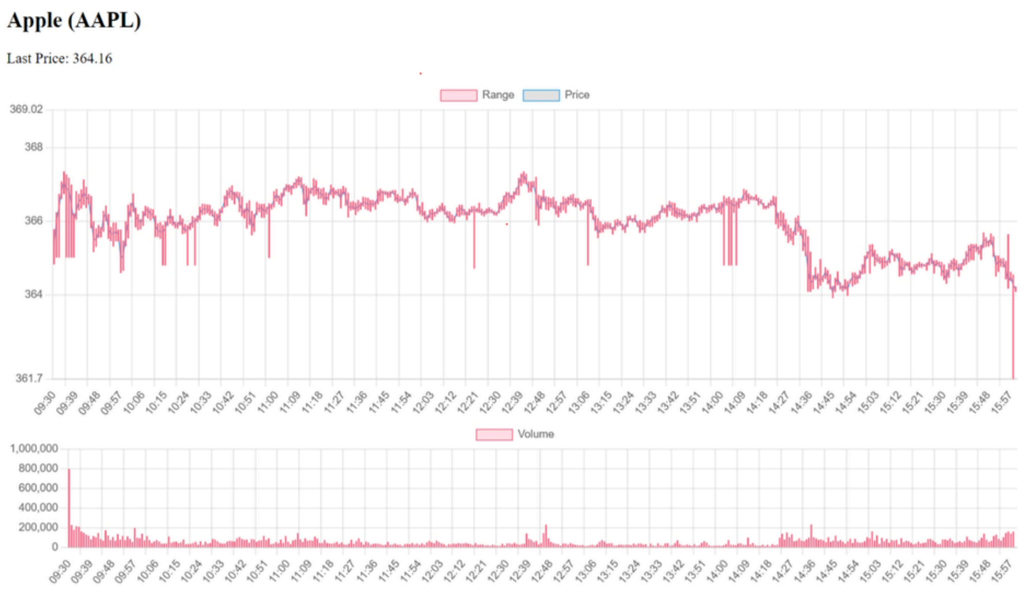
The browser should display the following:

- A graph displaying min and max price in each minute interval (floating bars)
- A graph display traded volume in each minute interval.
- The last price

If you are working in a group of three, then some additional analysis should be included in the final product.  This should include:

- Using candlestick graphs, instead of simple min/max
- Two Moving Averages and MCAD indicator
- Daily traded volume to date
- Opening price, day high, day low

Your final window should look something like this (only nicer!) and should have the additional information described for teams of three.



## Technology Requirements

You must use Gitlab throughout the project.  Regularly check in changes you have made to share with others in your team.  Your individual contribution will be assessed by the code you contribute in source control.

The technology you must use for the project is:

- FastAPI
- Websockets
- Pandas
- Graph.js

The application should be robust.  For instance, if the server is restarted, it should continue as before.  The browser should be able to reconnect at any time and reliably resync with the server.

Coding Guidelines:

- Code should be well structured and commented.
- Where appropriate classes should be used (e.g:  wherever functionality can be attached to data).
- Code should use exceptions where appropriate (particularly on connections).
- Code should include good logging.

## Considerations

Providing real-time data is not simple and ensuring system reliability is key - users must be able to depend on the data you publish and put in front of them.

There are a number of areas you will need to investigate in order to understand how to build this project.  For instance, you will have processes that are both publishing and receiving real-time updates, as well as service regular http requests.  You need to consider how this will be done.

For unknowns, during your design process you should consider building simple prototypes which demonstrate the basic functionality you need to build.  Even if you throw this away (and you should) it can significantly improve the speed and quality of your development.

To produce some of the real-time analysis you may wish to employ Pandas.  It can be helpful to work out the code to perform the translations on Jupyter first, before writing into your application code.

You will need to investigate documentation for the software components you are using - namely FastAPI, Chart.js and Pandas.  There is a wealth of information within the official documentation, you should practice being able to research and discover the capabilities of the packages you are using.

## Deliverables

You should deliver the following:

- All source code on Gitlab.  Also zip and upload Gitlab project to Moodle.  Include a ReadMe file which documents how to start and run the application for a user (e.g. Me).
- A document that contains a write up of your detailed design and implementation of the system.  Explain how each component of your system works and document any design decisions you made.

## Indicative Marking

The project is worth 25% of the total module mark.

Weighting of marks may vary a little for team size, but broadly indicative marking is as follows:

| | | |
|---|---|---|
| Simulator | Replay trades | 10 |
| | Publish RT data on API | 10 |
| | | |
| Server | Consume trade data | 10 |
| | Analyse data | 15 |
| | Publish RT data on API | 10 |
| | | |
| User Interface | Display price graph | 5 |
| | Display volume graph | 5 |
| | Display last, vol, high, low | 5 |
| | Update all in realtime | 5 |
| | | |
| Quality | Document | 10 |

| | | |
|---|---|---|
| | Code quality | 10 |
| | Proper use of source control | 5 |
| | | |
| | Total | 100 |