# Round 2 - Upstox FrontEnd Assignment

The objective of this assignment is to evaluate the skills of the candidate with a hands-on assignment.

Problem statement

Create a dashboard with the below two views.

1. **Home/Overview**

This screen should show a historical OHLC chart. You can get the historical data from the historical call in kaboom. The following list of metrics calculated from the historical data from kaboom.

a. Highest value and date
b. Lowest value and date
c. Start value and date
d. End value and date

2. **Live chart**

This screen should show a live OHLC updating chart by calling the subscribe method in kaboom. The chart should automatically get subscribed on opening the view and on change to a different view, it should get unsubscribed before the view changes. List of must have features:

a. You must be able to pan and zoom in the chart
b. The chart must be responsive.
c. The data must be stored so that if the user tries offline one should be able to view the chart. In case, internet is available, fetch the latest data.

## REST API

All REST API requests MUST be forwarded to the domain http://kaboom.rksv.net under the context /api hence **http://kaboom.rksv.net/api/**

**CORS** has been enabled at the Server and if you are facing CORS related issues, you are advised to use the CORS related standard HTTP headers.

**Debugging**

If needed, you can debug the API by visiting http://kaboom.rksv.net/ in your browser and see in action, how the communication is established.

Status

The Status REST API can be used to poll the API status. If the API Server is correctly configured and is running then this API will respond with a 200 OK and a JSON Object with the key status, whose value MUST always have the value OK.

Details

Method : HTTP GET

Context : /

Query String : none

Example
GET /api
STATUS: 200 OK
{
"status": "OK"
}

### Historical

The Historical REST API can be used to pull in historical records. The HTTP request can contain an optional query string interval whose value can range from 1 to 9. The interval will indicate the numbers of records to fetch, as described below

| Interval | Records Fetched |
|---|---|
| 1 | 200 |
| 2 | 400 |
| 3 | 600 |
| 4 | 800 |
| 5 | 1000 |
| 6 | 1200 |
| 7 | 1400 |
| 8 | 1600 |
| 9 | 1800 |

If the query string interval is not present or has value other than the one described above, then the entire dataset will be fetched, which contains about 2500 records.
The API will respond with a 200 OK and a JSON Array containing Strings of Comma Separated Values (CSV).
The CSV must be parsed as per the following header information **timestamp,open,high,low,close,volume,**
The timestamp is a Unix Epoch Timestamp.
The open, high, low, and close denotes the OHLC prices of that particular stock. Prices
must always be considered as floating point values.
The volume denotes the volume of trade conducted during that period of time.

### How to submit your assignment?

On completion of the assignment, you can share the project with us via any of the following:

1. Github/Bitbucket/gitlab link (Nice, atomic and iterative commits are encouraged)
2.Dont forget to include a readme with detailed instructions of how we can make it to work
3.Host your project on any free cloud hosting services so that we can see the app running. For example, you can use Heroku, Github pages or Surge.

## Guidance and expectations

1.We are keen to see how much you think is enough, and how much would go into a Minimum Viable Product. As a guide, elegant and simple wins over feature rich every time.
2.We are a brokerage firm, most of our developers and customers' lives revolve around charts, OHLC being one of them, so extra points for assignments that plot the data as OHLC charts.
2.Although the API might be returning relatively straightforward content, please try to structure your code as if you were building something more complex and production ready. We would like to gain an idea of how you would go about writing production ready code. So things like code structure, properly written test cases, a nice readme, deployment scripts etc would definitely add value to your code and its evaluation.
3.If you have any questions, feel free to ask. Points are not deducted for questions. Part of the exercise is to see how we communicate and if we can be efficient.
4.This challenge will be a critical component of your final round interviews with more members of the engineering team. It's fine if you take shortcuts and make assumptions - just be prepared to explain those to us while submitting the assignment and what you would have done if you'd had more time.