**GrabYo - Twitter Velocity Graph**

**Technology Choices**

You did say that the role is for a Java developer; however the assignment does state that I was free to use any technology or libraries of my choosing. As I am a few weeks into learning Python I opted to use it for my “back-end” solution as it would possibly make for a good learning experience (It did).

**Design and architecture**

Due to my inexperience communicating between the front-end and back-end my solution is a fairly simple affair. I opted to do so via data in a text file. The python writes all the relevant data and the JS reads and interprets it as it requires. I’m sure there are much more elegant solutions however I wanted to quickly prototype a solution that I know would work. I’ll be looking into these other solutions now. I’m imagining some form of an observer situation or DB storage so that the spike algorithm could be optimised through archived live event stream data. As it stands the backend script needs to be manually run however I envision that there could be a ‘master’ stream handler that would be aware of all required live events starting and ending times and it could start and stop many instances of this script as and when required.

**Backend code**

The backend is fairly straight-forward. We store the individual tweets as they come in. Then after every ‘tick’ (one minute) we count the number that occurred since the last tick. We then compare that value’s variance from the mean average of all of the tick tweet counts. If that variance is above a given threshold then a spike in the data is said to have occurred.

The latest tick’s values are written to a text file for the front-end to utilise. If a data spike has occurred then this is also communicated. Also if it was a spike tick then a few tweets will be written to a separate file for the front-end to display.

The only external library used was ‘Tweepy’ to simplify the Twitter APIs.

**Frontend code**

As it appears to be the case from your website the Grabyo studio video editor is a web-based platform so I opted for JavaScript to render the chart.

The key element of the front-end is the use of the chart.js library (<http://www.chartjs.org/>). Whilst it has many useful features I have also found it quite limiting for this particular scenario. E.g. odd mouse over animations that cannot be disabled and the inability to customise the visuals to look more like a feed of data rather than a chart. I’m looking into other data visualisation solutions.

There is also an additional feature that provides a selection of the latest tick’s tweets when a data spike has occurred. One use for this could be to help the end user identify the nature of the spike.

**Test cases**

Whilst I am comfortable testing with JUnit and Android there is currently no autonomous testing is in place for this project. I currently have no experience in testing with JavaScript or Python. I did however run a manual test for the Chelsea v Swansea game (25/02/2017) and the application correctly identified all the goals scored. This data is available in the attached ‘match\_data\_raw.txt’ file.

**Personal Note**

The assignment guidelines present this task as a chance to demonstrate my skills and knowledge. Whilst I‘m fairly happy with how the end-product shows off my skills (albeit for two fairly unfamiliar languages) I do however worry that it may also highlight the weaknesses I have with my knowledge of front-end and server-side development.

To counter this I would say that my strongest language by far is Java and I would be more than happy to rewrite the app written in a relevant Java framework. Furthermore if you would like to see some of my Java code I would suggest this recent Android project here… https://github.com/jaxfire/em\_project