# ElasticTweet

Joe Stoker

March 2018

## 1 Design and approach

A client-server model is used. The server serves an HTML, CSS and JavaScript application to the client. Jquery events are triggered by user interaction with the applicaion, which results in HTTP calls being made to a Python Flask[1] server. The Flask server then interacts with the Twitter API through the Tweepy[3] library, pulling Tweets based on a keyword specified by the client. These Tweets are then returned to the client in the HTTP response. A connection is established between the Flask Server and an ElasticSearch instance, so that Tweet data can be pushed to this instance. The Tweet data can then be viewed and analysed in Kibana.

## 2 Programming languages

The Shards[2] HTML and CSS template was used to style the application. JavaScript and Jquery were used to handle client side logic, as well as perform POST and GET requests. This created a smooth user interface and hence an overall better user experience.

Python was used as the back end language of choice, along with the Flask web framework. Flask is a micro-framework, designed for fast development of small web apps. Flask comes with a built in server, making development easier. While this server does not scale well, it is fine for this app that assumes a single user. The Python Tweepy library makes interacting with the Twitter API easier.

## 3 Obstacles

One of the obstacles faced was understanding how the Tweepy search API works. A search using a keyword only returns the Tweets that would appear on one page within Twitter. For this reason one needs to paginate across pages while searching, to get the correct number of Tweets.

Another challenge was understanding what ElasticSearch is, and how it is used with Kibana.

# References

[1] Flask. http://flask.pocoo.org/. [Online; accessed 1-March-2018].

[2] Shards. https://designrevision.com/docs/shards/. [Online; accessed 1-March-2018].

[3] Tweepy. http://www.tweepy.org/. [Online; accessed 1-March-2018].