CAB432 Assignment 2 Individual Report

Name: James Norman
SN: n10256989
Partner: Conor O'reilly

This report template uses similar conventions to those we used for Assignment 1 and the group task for Assignment 2. However, our requirements are more focused and there will be a strictly enforced limit of 3 pages - including diagrams and the heading and student information above. You should use only two diagrams – your overall assignment architecture and an alternative version suited to a global scale cloud application (we will provide more detail on this below). These diagrams should occupy no more than half a page in total, leaving one and a half pages for actual text. This is more than enough. The marking is based on the quality of the analysis and not on its length.

This individual task has two parts.

*In the first, we want you to analyse your assignment architecture with respect to statelessness, persistence and scaling. Note that this is not about **justifying** what you your understanding of these concepts. We expect you to look at your work with fresh, analytical eyes and to make some judgments about it. The marking is based on the quality of your arguments and not on how well you have designed and built the system. To answer this question you should draw on our coverage of statelessness, persistence and scaling in the unit materials since week 5. We expect that there will be some overlap between the analyses within a pair, but we will be marking them one after the other and the same marker will deal with both, so please remember that this must be individual work.*

In the second part we want you to consider your application and your architecture as it stands, and tell us how you would change the architecture to manage a high traffic, global version of the application in a similar domain, serving a similar purpose. This is a slightly artificial task, but here we want you to assume that the application does much the same thing, but that it does it with world-scale levels of traffic and data exchange. Here we expect you to supply a revised architectural diagram and to highlight the changes that become necessary at the revised scales. This question will draw further on your understanding of statelessness, persistence and scaling, but here we are especially interested in the principles discussed in Iain's lecture on Architecting for Cost.

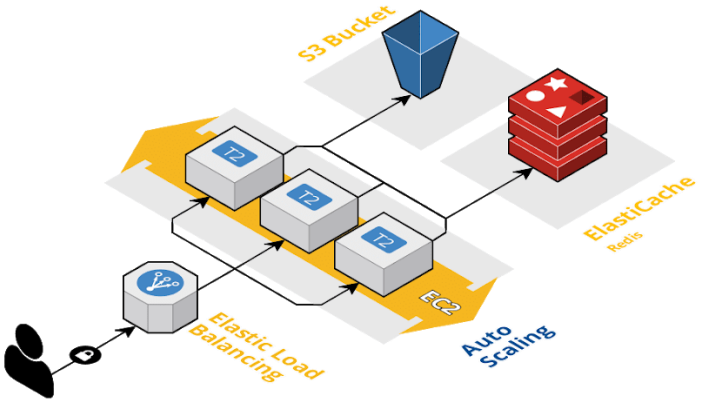
The different sections are discussed below. The task will carry a maximum of 20 marks and this will be scaled down to 10% of your final grade for the unit.

Statelessness, Persistence and Scaling (8 marks)

The application made has many correlations between the statelessness, persistence and scaling in regard to its architecture. The application has a good level of statelessness with the elasticache being in play through Redis, meaning that the data is there and is not stored locally. The s3 buckets also help with this. Meaning that if an instance fails the data is always backed up on the aws web servers.

Persistence has also been implemented fully into the application with the addition of the Redis cache and the s3 buckets with short term memory stored in Redis for faster access and long term memory stored in the s3 buckets when the Redis cache expires, the s3 buckets also provide the historical analysis in the application due to them being able to store information for a long time for a reasonable price unlike the Redis cache.

Scaling is the last topic to be covered here but is also done in a sufficient way with an autoscaling group set up in the aws to scale out based on CPU usage, with a target of 20% set, which is a perfect amount allowing a lot of CPU at any one time to scale all the way to 3 instances. Request count was looked at to be used. However, it was deemed not suitable for the application. There are draw backs with using CPU usage as the scaling parameter, as the application sometimes would not generate a high enough CPU load to warrant doing this. This was because of the free api's search limit and could have been used if the api allowed more then 100 queries a day.



The Global Application (12 marks)

To reimagine this application to a more global scale many things will need to be considered, the scaling in the aws would obviously have to be changed to support way more instances. The persistence choices may also have to change slightly, with the s3 cache for historical data not being a viable way to handle a ton of requests as the time it takes to return is ridiculous and would return high latency for users, a way to fix this would be to have historical data stored in Redis or somewhere like Cloudflare for the whole time, while this would prove costly It could be worth it for a better user experience. Apart from the s3 buckets latency I do not see any reason why the application would not scale to a larger pool. Eventual consistency would be a positive influence on the application as it would help store more historical data which whilst meaning the Redis cache has more data to store it would give the users a more in-depth insight into the historical sentiment. Additional cache levels would be used as the importance of edge cache cannot be underestimated, as it would provide a far better way of accessing data faster and more cost efficient. Scaling metrics would of course have to be changed to a higher CPU load. An upgrade to a higher power instance would also be needed as it could not run on a t2 micro unless a million instances were made. Ddos protection is also needed which can be done through a company like Cloudflare or amazon shield.

Due to the nature of this application there is very little reason for security concerns as I could not conjure a reason anyone would want to compromise this application. However, with that said that does not mean this application is safe as it is clearly not very well protected and could easily be hacked into or altered by someone capable. To mitigate this risk a website firewall would be implemented a security subscription would be kept a DDOS service would be used to protect from that and simple secure sockets would be implemented.