# CO3404 Distributed Systems Report

Yehan Banura Kapurubandara G20943498

## Critical Evaluation

How well I satisfied the requirements?

Firstly, the 40% marks requirement states to build and implement the Deliver Jokes microservice and Submit Jokes microservice, these two were initially created and ran on different port numbers. Afterwards, the “Submit Jokes” and “Deliver Jokes” was connected which allowed a user to input a joke into the “Deliver jokes” where it would be stored in the MongoDB Atlas database and then transferred to the “Deliver Jokes” which would allow the joke to be viewed. Following that, I created the “Moderate Jokes” service which has the functionality to read a joke directly from “Submit Jokes” and delete it from the “Submit Jokes” and store it in the “Deliver Jokes”. Furthermore, the ability to request a joke from the “Submit Jokes” edit that joke and submit it “Deliver Jokes” was implemented making the basic system complete. Also, there was a requirement which stated that “Submit Jokes” should be able to work even if the “Deliver jokes” microservice was down, I managed to achieve this by adding a feature which allows the “Submit Jokes” microservice to return a predefined set of types in case the “Deliver Jokes” microservice was not available. Then, the login portal was created for the “Moderate Jokes” which returns a bearer token that the user can use to access the complete functionalities that “Moderate Jokes” microservice has to offer however, without a bearer token none of the functionalities can be used.

Thereafter, I began the deployment process by initially deploying the “Submit Jokes” microservice on the Azure virtual machine, this was achieved by uploading my microservice to a git repository which allowed me to download it into the Ubuntu virtual machine hosted in Azure. Then the “Deliver Jokes” microservice was deployed in the same process however, I had to set up and configure mySQL in the virtual machine so my code can access it. After that, the “Moderate Jokes” needed to be uploaded into a docker container which I achieved by using some simple commands. Afterwards, it was pushed into an Azure registry container which allowed me to deploy the container instance of the “Moderate Jokes” microservice. Now that the three microservices are deployed and can be accessed from anywhere I began the documentation stage, this included making a swagger Ui interface that will allow users to see all the functionalities of the services.

Lastly, to achieve the 85% mark and higher I implemented SSL certificates to allow users to connect securely to the services and I created a frontend from Angular which showcases all the functionality of the microservices when combined. Eventually, I created a narrated video showcasing all the functionalities and implementations of the microservices. Therefore, I think the completed assignment overall belongs in the 85%-mark category

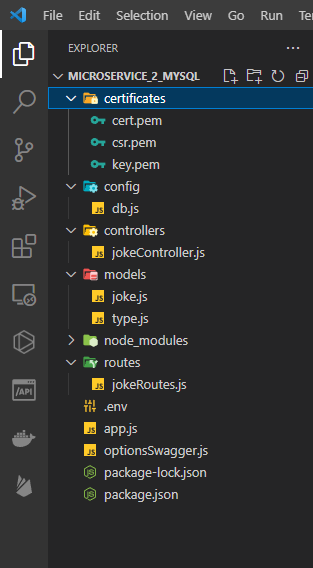
What areas can I improve?

Throughout the entirety of the assignment, I learned a lot of concepts and technologies which allowed me to understand what areas I lack. For example, it was challenging to use Linux when deploying the services and managing the SQL tables was also deemed very hard. I plan on improving my SQL knowledge and using Linux more often as it will be beneficial in the future.

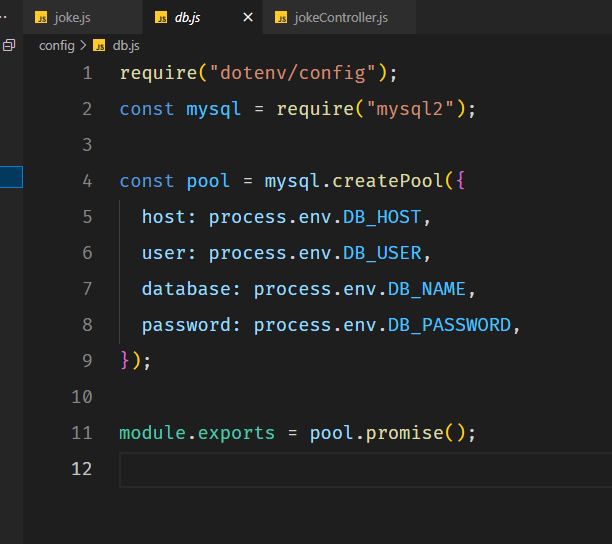
## Design patterns used

What design patterns have I used to provide screenshots and explain

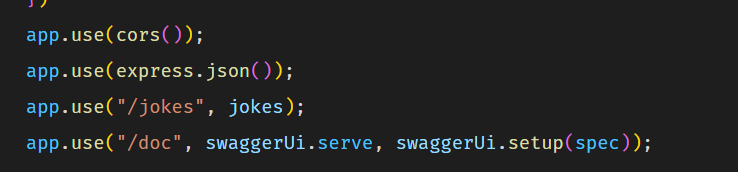
Firstly, the main design pattern which I used is the MVC (Model View Controller) Pattern which separates this application into three interconnected layers, these layers are the presentation layer, the business layer and the data layer. The below screenshot shows the structure which I have used for all my services.



Secondly, I have used the Singleton pattern which means that only one instance is created when making an object the below screenshot shows how that pattern was applied in the “Deliver Jokes” microservice.



Thirdly, I used the middleware design pattern which are functions that have access to the request and response objects which allows them to perform tasks like authenticating and handling errors.



Lastly, I think these design patterns were needed to make this project successful while being efficient and allowing the code base to be clear and understood by others who contribute

## Alternate Technologies

The technologies used in this project are quite popular and have a lot of communities that support them, the microservices were created by NodeJS which can be replaced by technologies like ASP.net which uses C#, and Java can also be used to create APIs. Furthermore, Google Cloud or Amazon Web services can be used instead of Azure cloud framework. We used Cloud services for our deployment model and there are other deployment methods like Container deployment which is used by Kubernetes, Serverless deployment, Hybrid deployment and more.