CHAPTER 1
INTRODUCTION

1.1. Background

In this rapidly growing world of information technology, the need to provide services to clients continues to increase, resulting in an application get too big in terms of a high number of lines of codes or a high number of people that use the application and a high number of people involved in the development of the application. Some programming languages that are widely used for the development of server side applications such as Python, C/C++ and Java are designed to be a single executable code that is called monolith. A monolith is a software application whose modules cannot be executed independently[4]. The monolith architecture has some drawbacks when it becomes a bigger application, when an application with many features gathered in 1 service it will be more difficult to maintain and the development speed will slow down. This is because the lines of code start to depend on each other, whereas changes and enhancements get more difficult. To solve these kinds of problems, an ideal architecture is needed. Johanes Thones states that microservices are: "a small application that can be deployed independently, scaled independently, and tested independently and that has a single responsibility"[11]. The definition that Johannes Thones stated addresses some characteristics of a microservices, such as microservice should be self-sufficient, flexible and fault tolerant. It also emphasizes that the machine should have a single responsibility.

The backend of the application that I create during the internship, uses a monolith architecture where all of the features are in 1 service, and on top of that the application will be used by many users and the application will be developed and updated with more features. As for today, the application is relatively small and does not have any major issue, but in the future the application will become big and it will create several problems, such as how fast the application will respond when there are a lot of users that use the application and in the development process where there is a feature that needs maintenance. The microservice architecture is designed to solve this kind of problems, the services that got separated will not depend on each other, so that when there is a lot of users accessing a specific service, the other service will not be disturbed, and in the development side the microservice architecture can be more resilient, in terms of when a

service goes down, the entire application does not stop functioning as it does besides the service that goes down.

The results of this project rely upon how big or small the microservice architecture will affect the backend performance than the monolith architecture. The benchmark of this project will rely upon how monolith architecture performs on the backend of the application. Both of the applications will undergo several load tests and will be compared.

1.2. Problem Formulation

This project focused on how to implement microservice architecture to a backend of the application and to see how it performs, is it better than monolith architecture that is already implemented or not. Specifically, it aims to answer the following questions:

1. How does microservice architecture average response time when there are a lot of users accessing the application compared to monolithic architecture?
2. Can the microservice architecture have a lower average response time than monolith architecture?
3. Does the amount of data affect both architecture performance?

1.3. Scope

The main limitation of this project is revolving on how the application performs after the microservice architecture has been implemented. The scopes described as follows :

1. The implementation and the development of the microservice architecture will only focus on the backend of the application.
2. The database of the application will use MySQL.
3. The main point of the comparison will focus on average response time.
4. The programming language that the application use is Go.

1.4. Objective

The purpose of this project is to seek for an alternative architecture to List Features Application backend, that previously used the monolith architecture, by using microservice architecture and to find out whether the microservice architecture can reduce the average response

time when there are a lot of users accessing the application compared to monolithic architecture and to describe the simplification process of the microservice architecture for handle the application even though there is a maintenance on a service.