CHAPTER 3
RESEARCH METHODOLOGY

3.1. Literature Study

This process is done by gathering and studying journals and research related to monolith architecture, microservices architecture which build upon various methods and are tested with various scenarios. Also studying a journal that discussed Go Language and the testing method. All journals studied were used as references for the project and program coded.

3.2. Building Monolith Architecture Web Server

This project aims to compare the performance between monolithic architecture and microservice architecture. In order to compare the architecture, the project needs a web server that is built upon both architectures. Since microservices were harder and more complex to build from scratch, it is easier to start the project by building a monolith application first. The application for both architectures has identical services so the comparison is objective. The monolith application is built on one PostgreSQL database. The monolith application consists of many services that shared a single database. Each service serves through API. The application runs as HTTP Server and all the services are accessible through one port.

3.3. Building Microservice Architecture Web Server

After building monolith application step is done, the next step is to build the microservice application. To build microservices that are based on monolith applications, the application needs to be broken down into smaller services. In this project, the application consists of merchant service, user service, and transaction service. Every service has its own database. Every service runs as HTTP Server and has a different port. Like the monolith, services are accessible by API and returning JSON response.

3.4. Setting Up NGINX Server

In the real world application, either front end or application users that consume API provided by the service use only a single HTTP port. Since the microservice application in this project has many ports running, it needs a load balancer and server proxy so that the service is

accessible from a single port. This project uses NGINX to do the server proxy. It is achieved through setting the NGINX config.

3.5. Testing

Both architectures were tested by using JMeter to hit the API provided by both applications. Since every service have unique features and yet both architectures have identical services, testing scenarios become various. The testing scenarios conducted are basic create, update, delete on merchant service, login service from the user service, and bulk transfer from transaction service.

3.6. Analyze

The testing result is latency and response time from the client-side (JMeter) and server log. This result was analyzed to compare the performance between both architectures to find out which architecture is better when handling scenarios given.