JIMP : Jurnal Informatika Merdeka Pasuruan
Vol.8 No.1 Maret 2023, P-ISSN : 2502-5716, E-ISSN : 2503-1945

Terakreditasi Peringkat Sinta 4 berdasarkan Petikan dari Keputusan Menteri Riset dan Teknologi/ Kepala Badan Riset dan Inovasi Nasional
Nomor 200/M/KPT/2020, masa berlaku mulai Vol.3 No.1 tahun 2018 s.d Vol.7 No.2 tahun 2022

Analysis of Speed and Connection When Accessing Information in Database Using Purposive Sampling Method

Muhammad Guruh Ajinugroho[1], Tubagus Maulana[2], Suryarini Widodo[3]
[1,2,3] Department of Computer Science, Faculty of Information Systems Management, Gunadarma University, Indonesia
[1]guruh.plsi.ug@gmail.com
[2]mkusuma@staff.gunadarma.ac.id
[3]srini@staff.gunadarma.ac.id

Received: 18-12-2022; Accepted: 15-04-2023; Published: 13-09-2023

Abstract— ***Animality Vetama is a pet store and clinic that has implemented an integrated management system which consist of several interconnected applications and a database across branches. Several branches complained about performances. Several factors was analyzed by the developer, resulting in a hypothesized that the problem was caused by connectivity issues on specific internet provider dubbed "provider X". Said problem is suspected to lies within the TCP port because it only occurs when accessing the system but not when doing other things like browsing. Client application connects to the database using MySQL direct connection. To prove this case, a study using purposive sampling quota approach was proposed to determine what makes a problematic provider. This approach was then followed by a simulation for each criteria by doing a test connection to the database using the existing method and a proposed method using webservice. The simulation were focused on average connection durations as well as success and failed ratio. The result shown that in general, connecting to the database using webservice is a lot time faster than connecting client directly regardless of the provider. The success ratio is also increased significantly.***

***Keywords*— web service, connection, internet provider, purposive sampling quota, troubleshooting, information system analysis and design, software engineering**

Abstrak— ***Animality Vetama merupakan sebuah toko hewan dan klinik yang telah menggunakan sistem manajemen terintegrasi yang terdiri atas beberapa aplikasi dan database yang saling terkoneksi antar cabang. Beberapa cabang mengeluhkan soal performa. Pihak developer melakukan Analisa terhadap beberapa faktor hingga terbentuk sebuah hipotesa bahwa permasalahan disebabkan oleh isu konektivitas penyedia jasa layanan internet tertentu yang dinamai "provider X". Permasalahan tersebut dicurigai terdapat pada port TCP dikarenakan hanya terjadi saat mengakses sistem namun koneksi lancar saat melakukan hal lainnya semisal browsing. Aplikasi klien terkoneksi ke database MySQL secara langsung. Studi kasus dilakukan menggunakan pendekatan purposive sampling quota untuk menentukan kriteria provider yang bermasalah. Pendekatan ini dilanjutkan dengan simulasi untuk setiap kriteria dengan melakukan uji koneksi antara klien dan database untuk kondisi yang berjalan saat ini, dan pada usulan untuk melakukan koneksi menggunakan webservice. Simulasi terfokus pada rata-rata durasi dan rasio keberhasilan. Hasil menunjukan bahwa melakukan koneksi ke database menggunakan webservice jauh lebih cepat ketimbang mengakses secara langsung. Rasio keberhasilannya juga lebih tinggi.***

Kata Kunci— ***webservice, koneksi, internet provider, purposive sampling quota, troubleshooting, analisis desain sistem informasi, software engineering***

I. INTRODUCTION

Today's technological development and information exchange have pushed everything to be entirely digital. In this all-digital world, existing technologies are required to always be reliable in serving and assisting humans in carrying out our daily activities. To achieve this, developers constantly strive to create innovations and solutions that suit customer needs. Animality Vetama, a pet store, and clinic is one of the businesses that has begun to adopt this digitization. This pet store consists of 17 locations operating in Depok City and Bogor Regency. As a clinic, each branch of Animality Vetama has practitioners and doctors who are certified experts in their fields, and every day they provide consultation and treatment for various types of animals, ranging from mammals such as cats, dogs, rabbits, and ferrets, to poultry animals and reptiles. As a pet store, this establishment offers a range of items for pets, including food, accessories, and grooming equipment. Some of these locations also offer medical treatment, boarding, and grooming services.

This store has implemented an integrated management system (IMS). This system comprises several applications that are linked to a server-side database. One of these applications is a desktop-based management program that is installed in each branch and used by retail staff to handle inventory management, sales, and purchase transactions, as well as reporting. Additionally, doctors and practitioners utilize this application to treat clinical patients, monitor patients, create prescriptions, and use other features related to veterinary clinics. Each branch is connected to the same database but has its unique code. There is also an android-based application that is used to report assets and total cash for each branch. These reports are processed by management applications of each branch, after which these reports are sent to the owners of each location and the central owner. Lastly, Animality Vetama has a website in the form of a landing page that serves as a corporate profile. However, some branches have reported that the desktop management application used in this system is not functioning optimally, mainly because of issues regarding the system's slow data transfer and access speed.

DOI http://dx.doi.org/10.51213/jimp.v8i1.783

According to the developer, the issue is frequently complained about by users when they use the sales transaction module. This module is one of the modules used in day-to-day business operations. However, additional information reveals that the suspected problematic cause, the internet connection, was able to run smoothly when used for activities other than using desktop management applications, such as browsing with a browser. When surfing the internet, the network is generally known to connect using the standard Hypertext Transfer Protocol port, commonly abbreviated as HTTP, namely port 80, or the standard port for Hypertext Transfer Protocol Secure (HTTPS), or port 443 [1]. This contrasts with the condition that applies to the desktop management application, where the application directly accesses the database on the server using the TCP MySQL port [2], [3], which is different from the port used when surfing in general. Allegedly, there are restrictions from the provider on connections that use ports other than 80 or 443. This notion is founded on three grounds. First, the developer discovered that store branches that were experiencing this problem were using the same internet provider. This problem was not reported by other branches that use different providers. Second, applications installed on each branch will always use the same version. Every time the developer updates the application's source code and uploads the update to the server, all branches receive an update at the same time. If a module in the application has a problem in one branch, it should also impact all other branches, which was not the case. Third, there are testimonies from other developers as well as articles [4] that highlight some of the shortcomings of the provider, which turned out to be the same provider used by each branch that experienced issues with access and data transfer speeds (the provider was disguised as "Provider X" in this study). One of the issues raised is the MySQL connection issue, which, according to sources, makes accessing the managed database difficult [5]. The suggested solution for this issue is to switch from Provider X's internet service to another provider. This is not feasible for the problems encountered in this study because not all Animality Vetama branch areas receive internet coverage areas from providers other than Provider X [6].

One possible solution is to modify how the desktop application connects to the database on the server by inserting one or more web services between the server and the location of the database, allowing the desktop management application to connect to the database later via the web service. However, before the development team can make a decision, it is necessary to test again whether there are indeed restrictions on the Provider X network and whether this is the cause of the desktop management application's slow data transfer speed.

In this study, an analysis will be carried out using a purposive sampling quota method by simulating to test port 80 connectivity as a port that is assumed to be safe for use on the Provider X network utilizing a web service and a TCP port used for MySQL network connections in desktop management applications by connecting directly to the database to determine the alleged limitation on port connectivity other than 80 on Provider X's network. Based on previous works in various disciplines, quota sampling in doing purposive sampling method could be used to define criteria for the participant if the research goals are known [7]–[10]. Simulations will also be performed on several branches using other providers as comparisons.

II. Research Methods

This section covers the methods used to do the research. The methods are divided into eight stages as shown in Figure 1.

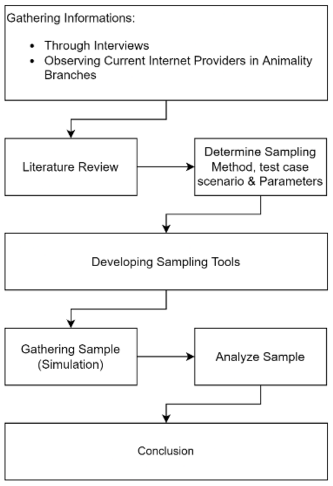

Gathering Informations:
Through Interviews
Observing Current Internet Providers in Animality Branches
Literature Review
Determine Sampling Method, test case scenario & Parameters
Developing Sampling Tools
Gathering Sample (Simulation)
Analyze Sample
Conclusion

Figure 1. Research Stages

A. Information Ghatering

Information is gathered to understand the current system's operating conditions. The development team of the Animality Vetama management system was interviewed for information. In addition to conducting interviews, the researchers observed each internet service provider (provider) used by branch users of the Animality Vetama management system's desktop management application. Based on the findings of this observation, the researchers learned that the Animality Vetama branch uses at least six different providers for daily work activities. Provider X, Biznet, Ayonet, JujungNet, Firstmedia, and one branch that uses XL cellular internet network tethering are among the six providers. These providers are listed as the members of Indonesian Internet Service Providers (APJII) [6].

B. Literature Study

The second stage is to conduct a literature review on previous research revolving around purposive sampling with a 5-year research time limit. Furthermore, the researcher conducted a literature review on the supporting theories that are relevant to this research.

C. Choosing a Sampling Method, a Test Scenario, and the Parameters to Use

The third stage involves determining the sampling method to be used based on the information gathered and the findings of the literature review about purposive sampling [9]. The quota method was chosen to carry out sampling because the purpose of the study was known, allowing the criteria required to group each participant to be determined [7], [8]. The purpose of the research is to compare data exchange performance between client user branches that use provider X network and user branches that use other providers when connecting directly to the MySQL database as the first test scenario, and then compare it to when connecting through a web service intermediary as the second test scenario.

There are two types of criteria used to categorize participants. To start, the criteria for providers are divided into problematic providers (group A) and other providers (group B) for comparison. Table 1 shows the divination of providers into two different groups.

TABLE I
GROUPING OF PROVIDERS INTO GROUP A AND GROUP B

Group A	**Group B**
Provider X	FirstMedia
	BizNet
	JujungNet
	AyoNet
	XL

Group A consists of two participant branches with the same provider, namely provider X, and Group B consists of five branches with different providers each. We divided and labeled each branch according to the provider group, the results are shown in Table 2.

TABLE III
GROUPING OF PROVIDERS INTO GROUP A AND GROUP B

Group A	**Group B**
Provider X	FirstMedia
	BizNet
	JujungNet
	AyoNet
	XL

In group B, each provider has one branch participant that is using them, thus we can just fit them directly into group B as B1, B2, B3, B4, and B5 respectively.

The parameters measured are the average duration of the information exchange process between the client and the server, as well as the data loss ratio during the process.

D. Developing Sampling Tools

At this point, the researcher creates a software-based test tool to collect samples from each branch that is the source of the data. Because each has a different working principle, two separate software or applications were created. The first app called DBConnMonitoring. It was used to test the first scenario. When simulating the exchange of information, this application monitors the connection. Figure 2 depicts the process of this application's working principle.

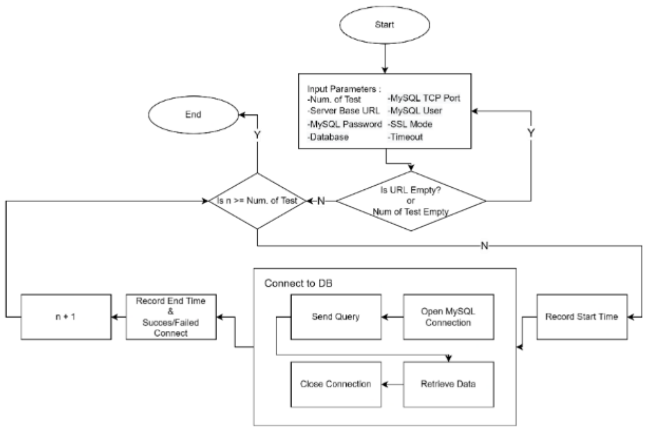


Figure 2. DBConnMonitoring Application Working Principle

As shown in Figure 2, First, the user fills each parameter. The parameters are described in Table 3.

TABLE IIIII
DBCONNMONITORING APPLICATION PARAMETERS

Parameter Name	**Function**	**Nullable**
Num. of Test	A number of n tests to be executed.	No
Server Base URL	Server's Public IP number.	No
MySQL TCP Port	TCP number used by MySQL.	Yes. Will have a value of 3306.
MySQL User	Username for MySQL.	Yes. Will use MySQL default installation user.
MySQL Password	Password for MySQL.	Yes. Will use MySQL default installation password.

Next, the user presses the "Execute" button, allowing the process to check the base URL and the number of tests; if any are missing, the process will not continue.

In the next flow, there will be a loop that checks the number of n that are currently being processed. n is a counter variable with an initial value of 0 that represents the number of tests (Num. of Test) that are currently running. If n equals or exceeds the number of tests, the sampling test is terminated. n will be incremented each time the test is completed In the next process, the program will record the Exec time. Exec time is the time when the application wishes to carry out a series of connection processes. Following that, the database connection is set up.

The application would attempt to create a MySQL connection using a connection string that was formed from the parameters in Table 3. Figure 3 is an example of a connection string :

```
server=111.123.4.222;port=3306;uid=root;pwd=pass;database=my
sql;sslmode=none;default command timeout=3600;
```

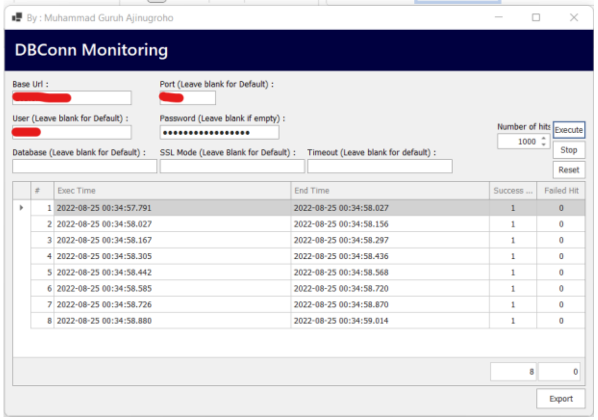
Figure 3. Example of a connection string in MySQL

After the connection is open, the program would send "SELECT VERSION()" query into the database. The command will return the current MySQL engine version that was installed on the server[11].

If the client successfully receives a response from the server, the application will record "Success Hit," otherwise it will record "Failed Hit." "Success Hit" and "Failed Hit" are data retrieved and data loss, respectively. It is important to note that the application will not consider whether the contents of the response match the query command on the request when determining the success or failure of a hit.

After that, the program records the end time. End time is the time after the program has done doing connection attempt in a sequence. Exec time and end time is having a format of yyyy-mm-dd HH:mm:ss.fff. yyyy represents the year in four-digit format, mm represents the month in two-digit format, dd represents a day in two-digit format, HH represents an hour in 24-time format, the second mm represents minutes in two-digit format, the ss represents seconds, and lastly, the mantissa of fff represent milliseconds. Here is an example of exec time or end time, 2022-08-25 00:34:57.791.

The program will then increment the number of variables n after recording the end date and time as well as the success or failure of the connection with the server. Figure 4 shows the user interface of DBConnMonitoring.



By : Muhammad Guruh Ajinugroho

DBConn Monitoring

Base Url : [redacted]

Port (Leave blank for Default) : [redacted]

User (Leave blank for Default) : [redacted]

Password (Leave blank if empty) : ••••••••••••••••••

Database (Leave blank for Default) :

SSL Mode (Leave Blank for Default) :

Timeout (Leave blank for default) :

Number of hits: 1000

#	Exec Time	End Time	Success ...	Failed Hit
1	2022-08-25 00:34:57.791	2022-08-25 00:34:58.027	1	0
2	2022-08-25 00:34:58.027	2022-08-25 00:34:58.156	1	0
3	2022-08-25 00:34:58.167	2022-08-25 00:34:58.297	1	0
4	2022-08-25 00:34:58.305	2022-08-25 00:34:58.436	1	0
5	2022-08-25 00:34:58.442	2022-08-25 00:34:58.568	1	0
6	2022-08-25 00:34:58.585	2022-08-25 00:34:58.720	1	0
7	2022-08-25 00:34:58.726	2022-08-25 00:34:58.870	1	0
8	2022-08-25 00:34:58.880	2022-08-25 00:34:59.014	1	0
			8	0

Figure 4. Example of a connection string in MySQL

Figure 3 also depicts when the DBConnMonitoring is running. The process will be terminated if the simulation reaches the number of Num. of Test (Number of Hits) or if the user presses the stop button and clicks yes on the pop-up that appears. Afterward, the table's sample data can be exported into a variety of file formats, including xls or xlsx, doc or docx, pdf, csv, txt, and image. Documents generated by "Export" can then be processed and used as analysis material as shown in Figure 4.

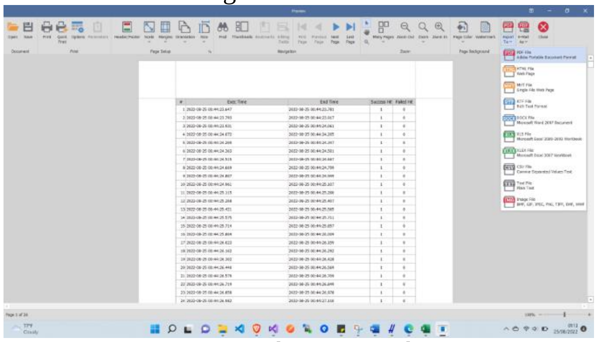


Figure 5. Generating document sample reports

The other testing scenario uses the modified version of DBConnMonitoring called DBConnMonitoringAPI. The key differences between DBConnMonitoringAPI with its counterpart are described in Figure 5.

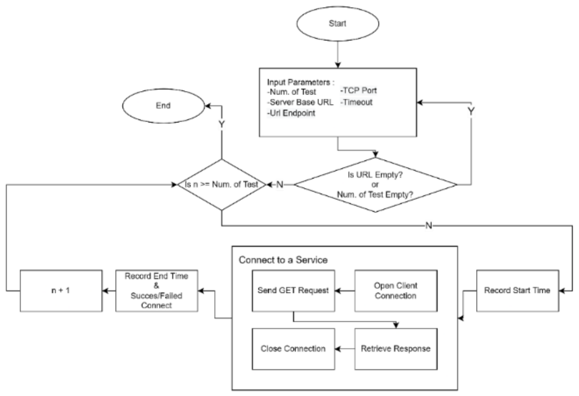


Figure 6. Working Principle of DBConnMonitoringAPI

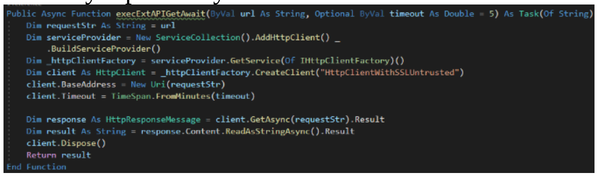
The application connects to a service rather than directly into the database after recording the start time begin. The parameters that are entered differ from those used by DBConnMonitoring. The parameters used in the DBConnMonitoring API application are listed in Table 4.

TABLE IVV
DBCONNMONITORINGAPI APPLICATION PARAMETERS

Parameter Name	Function	Nullable
Num. of Test	A number of n tests to be executed.	No
Server Base URL	Server's Public IP number.	No
TCP Port	TCP number for the destination service.	Yes. It will use value 80.
URL Endpoint	Targeted function in the service.	Yes. Will not be added to the URL.
Timeout	The amount of time the system waits for a response from the server's service. The time is measured in minutes.	Yes. Will use 300 seconds as the default value.

The Base Url, Port, and Endpoint parameters are then combined to form a URL that will be hit by the application using the REST GET method [12], [13]. An example of the resulting URL is as follows: http://111.112.3.332:80/service/fungsi.

DBConnMonitoringAPI uses the IHTTPClientFactory service provider to create a client based on parameters previously inputted by the user.



```
Public Async Function execExtAPIGetAwait(ByVal url As String, Optional ByVal timeout As Double = 5) As Task(Of String)
    Dim requestStr As String = url
    Dim serviceProvider = New ServiceCollection().AddHttpClient() _
        .BuildServiceProvider()
    Dim _httpClientFactory = serviceProvider.GetService(Of IHttpClientFactory)()
    Dim client As HttpClient = _httpClientFactory.CreateClient("HttpClientWithSSLUntrusted")
    client.BaseAddress = New Uri(requestStr)
    client.Timeout = TimeSpan.FromMinutes(timeout)

    Dim response As HttpResponseMessage = client.GetAsync(requestStr).Result
    Dim result As String = response.Content.ReadAsStringAsync().Result
    client.Dispose()
    Return result
End Function
```

Figure 7. Excerpts of code for function execExtAPIGetAwaitt

Figure 6 depicts execExtAPIGetAwait, a function used by an application to connect to and request a response from a web service. Within this function, an instance named serviceProvider is first created, which contains each service from the IServiceCollection. The function then declares _httpClientFactory as an IHTTPClientFactory object using the serviceProvider's GetService function. The client variable is then declared by the function as a HttpClient object created with the httpClientFactory instance and the HttpClientWithSSLUntrusted setting. The client instance will be converted into a HttpClient object, whose state will be monitored by IHTTPClientFactory and discarded when not in use [14]. The client is then configured based on the parameters entered by the user, such as BaseAddress, which is taken from the generated URL, and timeout, which is taken from the timeout parameter.

The client variable then sends an asynchronous GET request using the built-in function GetAsync, and the response is captured by a response object, which is an instance of HTTPResponseMessage. As the object is returned by the function, the response object's content will be retrieved and captured into the result variable asynchronously. If the response fails to capture feedback or does not receive a response from the server, the number of failed connection hits will increase.

Lastly, there is also a web service created for the purpose of doing simulation called dbconn_monitoring. This service act as an intermediary between DBConnMonitoringAPI and MySQL database on server. This service is an extension of Chris Kacergui's CodeIgniter-REST Server module version 3.1 [13], [15], which acts as an intermediary between the database on the server and the client in exchanging information [12]. The service is installed on the same server as the database so that it can access the database locally.

Lastly, there is also a web service created for the purpose of doing simulation called dbconn_monitoring. This service act as an intermediary between DBConnMonitoringAPI and MySQL database on server. This service is an extension of Chris Kacergui's CodeIgniter-REST Server module version 3.1 [13], [15], which acts as an intermediary between the database on the server and the client in exchanging information [12]. The service is installed on the same server as the database so that it can access the database locally.

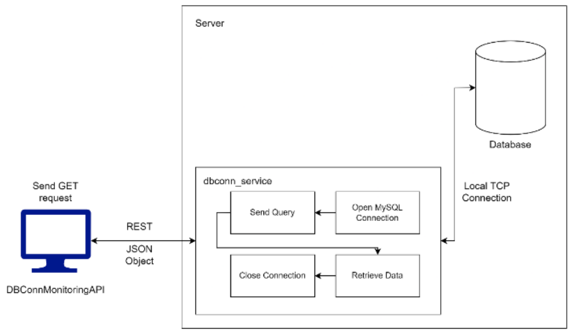


Figure 8. dbconn_service’s work scheme

Figure 8 depicts a web service that mediates the relationship between DBConnMonitoringAPI and the database via the REST API interface, where DBConnMonitoringAPI sends a GET request without a request body and header and receives a response in the form of a JSON Object.

E. *Simulation and Sampling*

In the next stage of the research methods, both the first and the second scenarios are running to collect the sample data on each participant.

F. *Data Sample Analysis*

The next stage is Data Sample Analysis. In this stage, the average duration and the average percentage of connection hit being a success or failure are calculated using formulas. The average duration is the time between the total end time. The average duration is used to calculate the time required to run a scenario simulation. Formula (1) is used to calculate the average duration.

$$\mu t = \frac{\frac{\sum t1}{\sum n1} + \frac{\sum t2}{\sum n2} + \cdots + \frac{\sum tx}{\sum nx}}{x} \quad (1)$$

μt is the average duration (ms), ∑tx is the total duration in x-number of simulations (ms), ∑nx is the number of samples in x-number of simulations. And x is the number of simulations used to test a scenario. The total duration is obtained using Formula (2).

$$\sum tx = t1 + t2 + \cdots + tn \quad (2)$$

The average percentage of "Success Hit" and "Failed Hit" is used to assess the dependability of each scenario in connecting and exchanging information. The average percentage of "Success Hit” is calculated using Formula 3.

$$\mu S = \frac{\sum s1 + \sum s2 + \ldots + \sum sx}{\sum n} \% \quad (3)$$

μS is the total percentage of “Success Hit” (%), ∑sx is the total of “Success Hit” in the x-number simulation. And ∑n is the total number of samples for all simulations in one scenario.

The average percentage of “Failed Hit” is calculated using Formula 4

$$\mu F = \frac{\sum f1 + \sum f2 + \ldots + \sum fx}{\sum n} \% \quad (4)$$

μF is the total percentage of “Failed Hit” (%), ∑fx is the total of “Success Hit” in the x-number simulation. And ∑n is the total number of samples for all simulations in one scenario.

III. Results and Discussions

In this section, a connection sampling test simulation is performed on the Animality Vetama branch according to the scenario defined in the previous chapter.

A. *Setup the Environment*

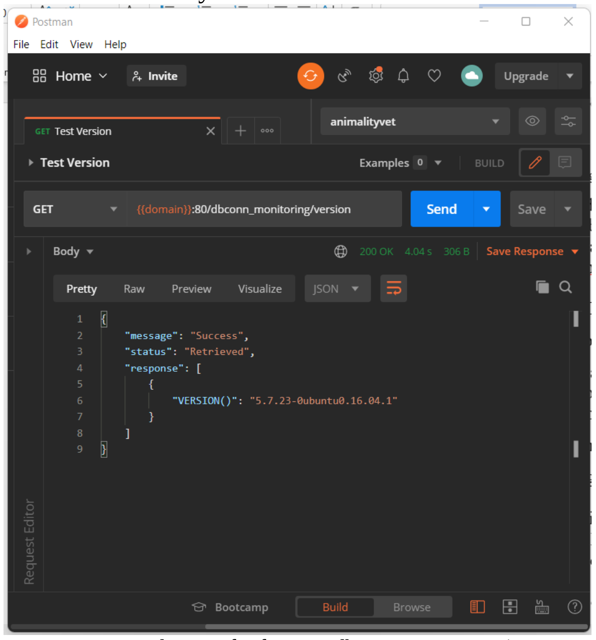
Before the test is carried out, the environment setup is first prepared by installing the pre-requisite software to do the test on a PC or laptop unit in the participant branches as defined in the previous section. Also, software [16] is installed on both researchers and branches to enable researchers to do the simulations remotely. The web service is also uploaded to Animality Vetama's server with the permission and help of the development team using an SFTP connection [17].

B. *Web Server Functionality Test*

After setting up the environment, a functionality test was carried out to ensure the dbconn_monitoring web service is able to respond to incoming requests from clients before doing the simulation. Postman [18] is used to carry out the test.

In Postman, the test environment could be configured. Dynamic variables could be added to store a value. When the environment configuration changes, the variable could be reused to store different values, as long as the same variable name exists in another environment setup. As an example, there are two postman environments created by users called animalityvet and localhost, and a variable {{domain}} is defined on both environments (env) and this {{domain}} variable is used as a base url that is a part of URI {{domain}}:80/dbconn_monitoring/version. On animalityvet env, it has a value of http://[server_ip (dubbed)] whereas, on localhost env, it has a value of http://localhost. So let's say the functionality test is handled using the localhost environment, then the {{domain}} would lookup into http://localhost, thus making the full URI to be http://localhost/dbconn_monitoring/version.

The environment is set to animalityvet. When called with the GET method via postman, the dbconn monitoring function returns response content in the form of a json object containing the parameter json object and a json array object containing the version of the MySQL database installed on the server. Figure 7 show an example of successful functionality test when we hit {{domain}}:80/dbconn_monitoring/version using the environment animalityvet.



```
GET {{domain}}:80/dbconn_monitoring/version
200 OK 4.04 s 306 B
{
    "message": "Success",
    "status": "Retrieved",
    "response": [
        {
            "VERSION()": "5.7.23-0ubuntu0.16.04.1"
        }
    ]
}
```

Figure 7. Functionality test for function dbconn_monitoring/version

C. *Sample collection*

After setting up the environment, a functionality test was carried out to ensure the dbconn_monitoring web service is able to respond to incoming requests from clients before doing the simulation. Postman [18] is used to carry out the test.

Data sampling is performed in each branch of the Animality Vetama clinic by running simulations for retrieving information directly from the MySQL database using DBConnMonitoring and retrieving information via the web service using DBConnMonitoring API, with each test being performed 5 times for each participating user branch. In each test, connections were made 1000 times sequentially. Each branch's tests are run in parallel. The reason for this is to determine the server's performance when dealing with a large number of requests at once. Table 5 shows some sample data from one of the participating branches.

TABLE V
EXAMPLE OF SAMPLE DATA FROM ONE OF THE BRANCHES

#	Exec Time	End Time	Success Hit	Failed Hit
1	2022-08-24 22:14:17.484	2022-08-24 22:14:17.599	1	0
2	2022-08-24 22:14:17.599	2022-08-24 22:14:17.634	1	0
3	2022-08-24 22:14:17.650	2022-08-24 22:14:17.686	1	0
...				
1000	2022-08-24 22:14:17.758	2022-08-24 22:14:17.799	1	0
			1000	0

The exec time refers to when the application tries to connect, and the end time is when the connection is completed. Success Hit indicates a successful connection and server response, whereas a Failed Hit indicates the opposite. The following is the result of the simulation conducted on each provider in each branch. Table 6 shows an example of sample data with an example if it fails to connect.

TABLE VI
EXAMPLE OF SAMPLE WITH FAILS

#	Exec Time	End Time	Success Hit	Failed Hit
1	2022-08-24 06:21:43,781	2022-08-24 06:21:43,805	1	0
2	2022-08-24 06:21:43,809	2022-08-24 06:21:43,845	1	0
3	2022-08-24 06:21:43,857	2022-08-24 06:21:43,883	1	0
4	2022-08-24 06:21:43,889	2022-08-24 06:21:43,908	1	0
...				
271	2022-08-24 06:21:55,073	2022-08-24 06:21:55,084	0	1
...				
863	2022-08-24 06:22:23,073	2022-08-24 06:22:23,097	0	1
...				
997	2022-08-24 06:22:30,873	2022-08-24 06:22:30,890	1	0
998	2022-08-24 06:22:30,905	2022-08-24 06:22:30,922	1	0
999	2022-08-24 06:22:30,937	2022-08-24 06:22:30,957	1	0

…				
			998	2

D. Obtaining Average Duration of Simulation

Based on the formula that has been described in chapter 3, the average duration is calculated using the defined formula stages. Below is an example of the steps on calculating the simulation average duration for branch A1.

First, the difference between "Exec Time" and "End Time" in the 1st data of simulation 1 for the first scenario is calculated as shown in Formula 4 and 5.

$$t1 = 2022-08-24\ 20{:}08{:}23{,}004 - 2022-08-24\ 20{:}08{:}21{,}450 \quad (5)$$

$$t1 = 1{,}554\ (s) = 1554\ (ms) \quad (6)$$

The first step is then repeated until the duration of each data is obtained. Second, All durations in simulation 1 were calculated using Formula 2 into the total duration.

Lastly, each total duration from 5 simulations is calculated using by dividing each by their n sample (we use 1000 for each simulation) and divide them by the x-time simulations was being conducted. The result is shown in Formula 6 and 7.

$$\mu t = \frac{\frac{182397}{1000}+\frac{188969}{1000}+\frac{177092}{1000}+\frac{149377}{1000}+\frac{172335}{1000}}{5} \quad (7)$$

$$\mu t = 174{,}034\ (ms) \quad (8)$$

Table 7 is the result of calculating the average duration of the simulation for the first and second scenarios in each branch.

TABLE VII
THE AVERAGE DURATION IN THE FIRST SCENARIO AND THE SECOND SCENARIO

Branches	**µt First Scenario (ms)**	**µt Second Scenario (ms)**
A1	174,034	1674,254
A2	35,3614	591,1446
B1	45,3998	139,8064
B2	204,8838	118,6212
B3	47,2128	233,9742
B4	181,7812	306,8874
B5	66,7468	273,6784

Based on the duration calculation results in Table 7, it can be seen that the µt of the first scenario is always smaller than the µt of the second scenario in the majority of the participating branches. In the case of branch B2, the first scenario's µt is actually greater than the second scenario's µt.

E. Obtaining Average Percentages of "Success Hit" and "Failed Hit"

Based on the sample data that has been collected in each branch, the average percentage calculation for "Success Hit" and "Failed Hit" is calculated. The following is an example of calculating the average percentage of "Success Hits" and "Failed Hits" in the first scenario for participant branch A1. Formula 8 and 9 is an example of calculating the percentage of "Success Hits" from all simulations.

$$\mu S = (1000+1000+1000+1000+1000)/5000\ \times 100 \quad (9)$$

$$\mu S = 100\% \quad (10)$$

Formula 10 and 11 is an example of calculating the percentage of "Success Hits" from all simulations.

$$\mu F = (0+0+0+0+0)/5000\ \times 100 \quad (11)$$

$$\mu F = 0\% \quad (12)$$

Table 8 shows the results of calculating the average percentage of "Success Hit" and "Failed Hit" in the first and second scenarios.

TABLE VIII
AVERAGE PERCENTAGE OF "SUCCESS HIT" AND "FAILED HIT" IN THE FIRST AND SECOND SCENARIO

Branches	µtS(%) First Scenario	µtF(%) First Scenario	µtS(%) Second Scenario	µtF(%) Second Scenario
A1	100	0	99,96	0,04
A2	99,9	0,1	99,34	0,66
B1	100	0	100	0
B2	99,94	0,06	99,98	0,02
B3	100	0	100	0
B4	100	0	100	0
B5	100	0	99,96	0,04

According to the results of the average percentage calculation in Table 8, the first scenario has a "Success Hit" percentage of 99.9 to 100 percent. Meanwhile, if you run the second scenario simulation, the average "Success hit" percentage ranges from 99.34 to 100 percent. This demonstrates that the first scenario increases the likelihood of a successful connection when accessing or exchanging data with the database.

IV. CONCLUSION

The simulation testing and comparing direct connection to the database with a connection via a web service is carried out through a series of stages and using the purposive sampling quota method to determine provider criteria to pick branches to be participants. the connection test is run on each branch using two different software to obtain samples for each connection scenario through five time simulations. The sample is processed and analyzed by calculating the formula to determine the average duration of the simulation for each scenario and the formula to determine the percentage of "Success Hit" and "Failed Hit" in the overall simulation for each scenario. The result shows several thing regarding the allegation that provider X has connection restrictions on ports other than 80. Connection directly to the database took 9 to 16 times longer than connecting via web service, it could reach up above 1500ms three times higher than other branch which only took 300ms. Furthermore, in 5 simulations where the database is accessed directly, the percentage of successful connections is never 100%. This condition is demonstrated in all participating branches that use provider X. The result shows that in most cases, connecting to a database to exchange information through a web service is preferable to connecting between applications and the database directly

The analysis of the sample data in this study is limited to determining the average duration of data exchange and the percentage of success and failure. Furthermore, when simulating information exchange with a database via a web service, the port connection compared in this study is only limited to port 80. Branches representing each provider and provider X were still limited. Additional research can be conducted to ensure the accuracy of the data sent by the server in response to the client's request. The researcher also hoped that further research could include a variety of providers and branches to represent each provider variety, as well as ports that are checked during scenario simulation.

REFERENCE

[1.] Aakanksha, B. Jain, D. Saxena, D. Sahni, and P. Sharma, “Analysis of hypertext transfer protocol and its variants,” Advances in Intelligent Systems and Computing, vol. 670, pp. 171–188, 2019, doi: 10.1007/978-981-10-8971-8_17/COVER.

[2.] MySQL, “MySQL :: MySQL 8.0 Reference Manual :: 4.2.7 Connection Transport Protocols.” [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/transport-protocols.html.

[3.] MySQL, “MySQL :: MySQL 8.0 Reference Manual :: 4.2.4 Connecting to the MySQL Server Using Command Options.” [Online]. Available: https://dev.mysql.com/doc/refman/8.0/en/connecting.html.

[4.] INSPIRASIBIZ, “Akses Program dari Internet Menggunakan koneksi Indihome Fiber,” INSPIRASIBIZ, 2017. [Online]. Available: https://inspirasi.biz/knowledgebase/detail/KB-20171013111031/KBKAT-20170403132947.

[5.] Setiawan, “Layanan Indihome yang Kurang Bersahabat bagi Developer — Arif Setiawan,” Personal Blog, Jan. 2017. [Online]. Available: https://arifsetiawan.com/2017/01/layanan-indihome-yang-kurang-bersahabat-bagi-developer/.

[6.] APJII, “Daftar Anggota Penyelenggara Jasa Internet Indonesia (APJII).” [Online]. Available: https://apjii.or.id/anggota/index.

[7.] J. Wood, M. Graham, V. Lehdonvirta, and I. Hjorth, “Good Gig, Bad Gig: Autonomy and Algorithmic Control in the Global Gig Economy,” Work, Employment and Society, vol. 33, no. 1, pp. 56–75, 2019, doi: 10.1177/0950017018785616.

[8.] S. I. Fannani, M. Najib, and M. Sarma, “THE EFFECT OF SOCIAL MEDIA TOWARD ORGANIC FOOD LITERACY AND PURCHASE INTENTION WITH AISAS MODEL,” Jurnal Manajemen & Agribisnis, vol. 17, no. 3, pp. 285–285, Nov. 2020, doi: 10.17358/JMA.17.3.285.

[9.] S. Campbell et al., “Purposive sampling: complex or simple? Research case examples:,” https://doi.org/10.1177/1744987120927206, vol. 25, no. 8, pp. 652–661, Jun. 2020, doi: 10.1177/1744987120927206.

[10.] M. A. P. Bahinting et al., “Stronger than the Internet Connectivity: A Phenomenology,” Jul. 2022, doi: 10.5281/ZENODO.6791820.

[11.] Tondak, “A Beginner’s Guide to SQL Commands: DDL, DML, DCL, & TCL,” Microsoft Azure Data Engineer Certification [DP-203] & Q/A, 2022. [Online]. Available: https://k21academy.com/microsoft-azure/data-engineer/sql-commands/.

[12.] Halili and E. Ramadani, “Web Services: A Comparison of Soap and Rest Services,” Mod Appl Sci, vol. 12, no. 3, p. 175, Feb. 2018, doi: 10.5539/mas.v12n3p175.

[13.] F. Herdiyatmoko and Y. D. Pratama, “IMPLEMENTASI RESTFUL SERVER MENGGUNAKAN LIBRARY CHRISKACERGUIS CODEIGNITER 3,” Seminar Nasional Informatika (SEMNASIF), vol. 1, no. 1, pp. 1–7, Dec. 2020, [Online]. Available: http://103.23.20.161/index.php/semnasif/article/view/4079.

[14.] K. Larkin, S. Gordon, G. Gordon, and R. Nowak, “Make HTTP requests using IHttpClientFactory in ASP.NET Core | Microsoft Docs,” 2022. [Online]. Available: https://docs.microsoft.com/en-us/aspnet/core/fundamentals/http-requests?view=aspnetcore-6.0.

[15.] Kacerguis, “CodeIgniter RestServer,” Github, 2019. [Online]. Available: https://github.com/chriskacerguis/codeigniter-restserver.

[16.] AnyDesk, “About Us – AnyDesk,” 2022. [Online]. Available: https://anydesk.com/en/company.

[17.] Martin, “Integration with PuTTY :: WinSCP,” WinSCP, 2022. [Online]. Available: https://winscp.net/eng/docs/integration_putty.

[18.] Postman, “Postman API Platform,” 2022. [Online]. Available: https://www.postman.com/product/what-is-postman/.