Applying Integrating Testing of Microservices in Airline Ticketing System

Dearisma Arfinda Ma'ruf[1], Selo Sulistyo[2], Lukito Edi Nugroho[3]

Abstract—**Microservices have been applied on several major systems including airlines. The characteristics of microservices which are independent and also interconnected need to be tested. The testing is done to preserve the system's sequential stage processes, especially the online ticket reservation. Four features which are the search, booking, payment, and booking info feature are tested. This research performed three stages of testing on the microservices, those are unit testing, integrity testing, and end-to-end testing. Unit testing was conducted to test every function on every nodule, integrity testing was done to test interconnection between microservices, and end-to-end testing was to test the final results obtained after the unit test and integrity test were carried out. The three stages of testing must be done sequentially. The system on the airline provides the valid or correct response. Three stages of testing can be applied on other airlines by obtaining a legal API and can be accessed publicly.**

Keywords—**Microservices, Unit Testing, Integrity Testing, End-to-End Testing.**

I. INTRODUCTION

Microservice is a collection of independent processes that communicate to each other in order to form a complex application towards any API language or commonly referred to Application Programming Interface [1]. API is a platform which bridges users and the information system [2]. Services in a microservice can be easily controlled in its development. Microservice is able to divide an application into several smaller services which are connected to one and another.

The architecture of microservice is different from other forms of architecture. The microservice form of architecture is more structured and flexible as an alternative for others [3], [4]. Information system in the architecture of microservice is more distributed and it can provide more focused and detailed services. Problems in microservices are divided into smaller parts (solution) which are arranged into one service and each service has its own role and responsibility. From this definition, a system or software that applies the microservice consists of several or many services that are managed and distributed independently, so that it allows the system to easily adapt to the changing needs.

The main challenge for the microservice is to conduct the test on the microservice which has independent and interconnected characteristics. Stages of testing on a microservice are different from testing on a system that do not implement a microservice. The testing on the microservice must be conducted gradually since microservice has a characteristic which is connected to each other. There are several things which have to be done in testing the microservice such us unit testing and final testing. Besides, the main challenge is to test the dependency between the microservices in order to know whether or not the results of one microservice has the dependency on the results of other microservices. The microservice testing is done in several stages including unit testing, service test, end-to-end test, behavior-driven test, integration test, and regression test [5]–[7].

There are few researches that discuss the microservice testing. Those researches only briefly discuss the stages of microservice testing. For example, the stage of testing on the microservice is performed with a regression test that compares the response output of the developing system with the running system. This stage is applied in the National Iranian Oil Products Distribution Company Sales System [7].

In another research, the stage of testing on the microservice is behavior-driven test. It describes a person's behavior that is expected from a feature and criteria in the form of a simple scenario so that it can be understood by common people [1]. This research discusses mechanisms testing for microservices without discussing the details of the testing stages [5].

Thus, this research proposes three stages of testing on the microservice and these stages must be carried out in sequence. This research explains one by one the testing stages used and applies the integrity testing stages to test dependencies between microservices. Those stages of testing are unit testing, integrity testing, and end-to-end testing.

In Indonesia, several well-known companies implement the microservice in their business systems. One of them is Sriwijaya Air which is one of Indonesia's largest airlines and one of the Indonesian domestic leading airlines. It carries over 950,000 passengers per month from its hub at Soekarno-Hatta International Airport to 53 destinations within Indonesia and three regional countries including extraordinary and popular tourism spots in Indonesia and regional countries. Established on November 10th, 2003, Sriwijaya has several vision and mission, one of them is delivering high quality services. Hence, Sriwijaya Air has already implemented API that is still under development in their flight systems [8]. Therefore, this research uses Sriwijaya Air as a case study and this research implements three stages of microservice testing on the API which is being developed by Sriwijaya Air.

II. TESTING STRATEGY

The testing was done by implementing a testing strategy. The testing strategy was able to work by providing several test cases or test scenarios. The test case was used to determine the testing. An unsuccessful test case was the cause of a program

[1,2,3] *Department of Electrical and Information Engineering, Faculty of Engineering, Universitas Gadjah Mada, Jln. Grafika No. 2, Kampus UGM, Yogyakarta, 55281, INDONESIA (Tel. +62-274-552305, email: [1]dearisma.arfinda.m@ugm.ac.id, [2]selo@ugm.ac.id, [3]lukito@ugm.ac.id)*

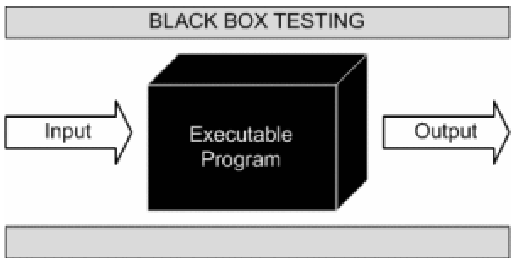


Fig. 1 Black-box testing.

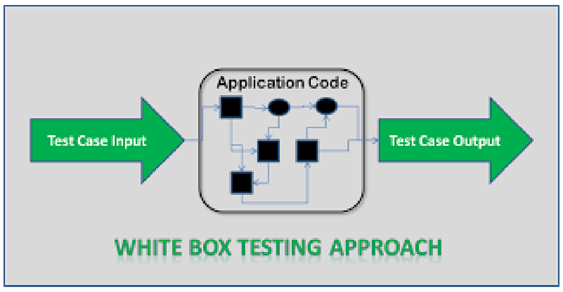


Fig. 2 White-box testing.

to produce correct results without finding any errors. Two of the most prevalent strategies including black-box testing and white-box testing will be explained below [9].

A. *Black-box Testing*

Black-box testing focuses on the results of the execution through data test and functional checking of the application that is being tested. This test is also known as data-input or input-output testing. The advantage of the black-box testing method is that the testing is done without the need to have knowledge of the application programming language.

This test is done based on the user's needs, so that if it does not match the user's needs or the predictions, it can be easily corrected. This test is carried out by utilizing every possible input condition as a test case. The purpose of this test is to find circumstances where the program does not behave according to its specifications. The overview of black-box testing strategy can be seen in Fig. 1.

B. *White-box Testing*

White-box testing is often called logic-driven testing. White-box testing method emphasizes the testing based on the source code of the program created. The testing is used to find bugs in the code, typos that occur when writing the code, and assumption errors in programming. Thus, errors can be fixed at the same time. There are several approaches used in the white-box testing method or technique which are carried out. Therefore, the white-box testing is called structural testing. An overview of white-box testing strategy is shown in Fig. 2. The comparison and the details about black-box testing and white-box testing strategies can be seen in Table I.

TABLE I
THE COMPARISON OF THE TESTING STRATEGIES

No	Black-box Testing	White-box Testing
1	The internal workings of the application are not required to be known	The tester has full knowledge of the internal workings of the application
2	Also known as closed box testing, data driven testing and functional testing	Also known as clear box testing, structural testing and code based testing
3	Performed by end users and also by testers and developers	Normally done by testers and developers
4	• The testing is based on external expectations • Internal behaviour of the application is unknown	Internal workings are fully known and the tester can design the data test accordingly
5	Time consuming and exhaustive	The most exhaustive and time consuming type of testing
6	Not suitable for algorithm testing	Suitable for algorithm testing
7	This can only be done by trial and error method	Data domains and internal boundaries can be better tested

III. MICROSERVICES TESTING STAGES

This research proposed three stages of testing which were performed manually and utilizing the Postman tool. Each stage of testing had to be done in sequence from beginning to the end with the appropriate input. The three testing stages are explaind as follows.

A. *Unit Testing*

Unit testing stages were required to test every small unit components in the system. This process was done to ensure the functions in the system worked properly. This research performed microservices testing of software functionality. The testing was done by giving proper inputs according to the obtained structure of API. The inputs which were usually given were text, number, time, and date. This testing was aimed to verify the inputted parameter and retrieved the correct value [9], [10].

B. *Integrity Testing*

The integrity testing stage was required to check the performance of small unit components worked as a combination and not as an individual unit. The data used was remain the same in the form of text, number, time, and date. After the unit testing was done, the next step was to test the relationship between microservices by maintaining the sequential microservice. This stage was aimed to know whether the process was carried out in order of each microservice or it was done randomly. In addition, this stage needed to be conducted because it was based on the characteristics of the microservice which was still related to one another although they were independent [10], [11].

C. *End-to-end Testing*

From the previous two stages of testing, the end-to-end testing was performed to see the final result using the system. This stage focused on whether the given input in the unit testing and integration testing had entered the system or it had been

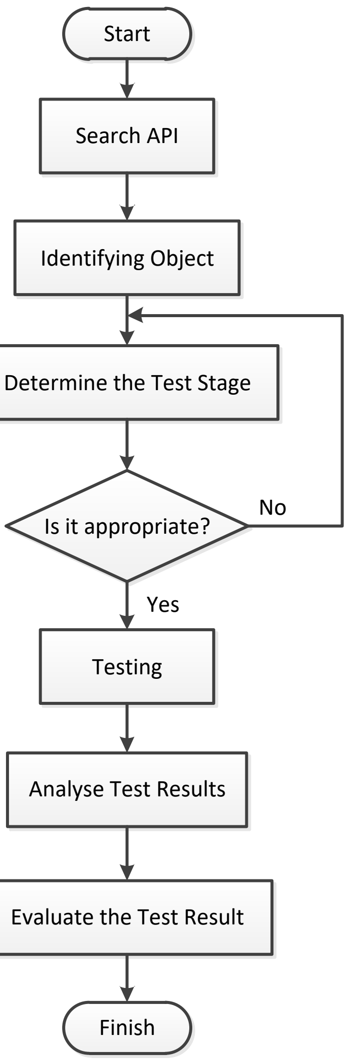


Fig. 3 Testing flow.

stored. In this research, the testing metric was done by looking at the final result of the test [10], [11].

IV. Testing Results

The test is carried out by applying the testing stages which is shown in Fig. 3. There are several stages of testing on the microservice and it can be seen in Table II.

Those stages of microservices testing show the advantages and deficiencies of each testing stages. Three methods which are unit test, integrity test, and end-to-end test have the advantages that can be easily applied. These three methods are sufficient to represent several other methods. It is because they tested the smallest functions individually, the service between microservices, and even the overall system result. These three stages are represented from the smallest functional stages to test the whole system. These three stages are not only easy to do, but also have short estimation of time. Besides, they have low cost which is important. Developers can be assisted with three stages of this testing method. In the implementation of testing, there are 13 variables for the test scenario. Some scenarios are shown in Fig. 4.

Table II
Stages of Microservices Testing

No	Stages	Deficiency	Advantages
1	Unit Testing	-	Easy to use Low cost
2	Integrity Testing	-	Easy to use Low cost
3	Component Testing	Expensive cost	Focus more on testing
4	Contract Testing	Longer estimated time	-
5	End-to-end Testing	-	Easy to use Low cost
6	Performance Testing	Longer estimated time	More detail on the testing

Based on the test, it is revealed that Sriwijaya Air’s API only implements four features into the plane ticket reservation, which are explained as follows.

1) Search: A feature in the plane ticket reservation which the function is to search a flight.

2) Book: A feature in the plane ticket reservation that is used after the search process is completed. Booking is a process of ordering and filling in the details of the passengers by submitting the contact person and the names which are categorized (either as adults, children, or toddlers).

3) Payment: A feature in the plane ticket reservation which is functioned as a way to conduct payment and is done through Automated Teller Machine.

4) Book Info: A feature in the plane ticket reservation to get the booking code after the previous processes.

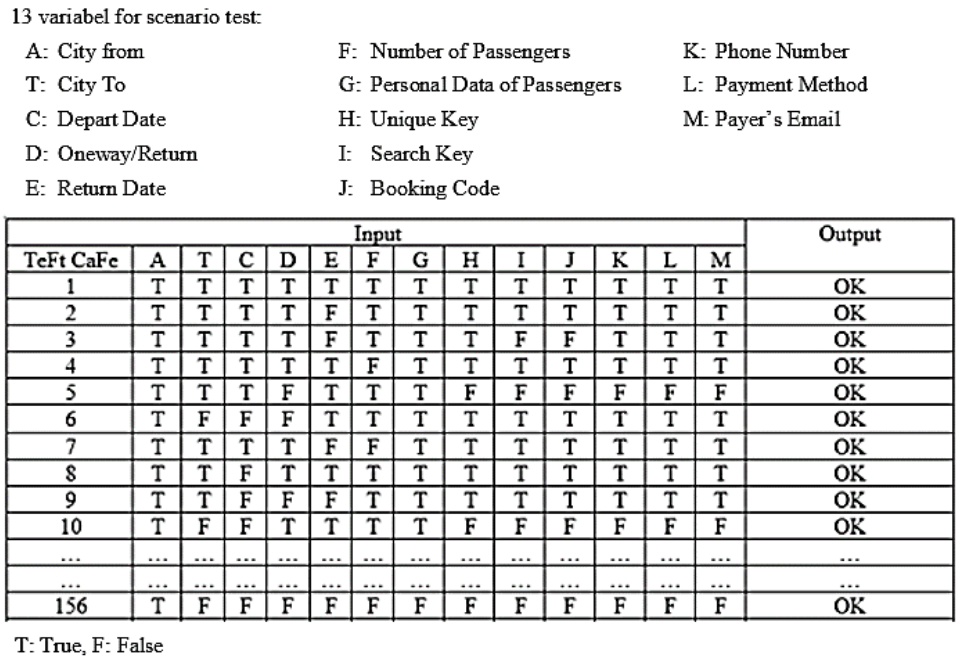
From the limited observation above, it is shown that the stages are implemented with the three stages of tests that lead to the following results.

A. Unit Testing

In the unit testing, the test was performed by testing every microservice such as the search feature, book feature, payment feature, and booking info feature. In this stage, the testing used the metric to make sure that the inputs and outputs were as expected. The inputs were done according to the API structure.

1) The Implementation on Search: This stage was conducted using the search feature containing Search Request and Search Response. Search Request was used to request flight schedules and Search Response is used to give response by displaying the return data of flight schedule’s details. Further information is presented in Fig. 5.

2) The Implementation on Book: This stage concerned with the implementation of the book feature containing Book Request to request passengers’ details. The inputted data must be about the journey (flight information), paxCount (total passenger), contact (contact information), paxs (the details of passengers’ names), and itineraries (the search key from Search Response). The function of the book response is to respond to the request by displaying data on reservation details in a form of passenger detail and flight detail.



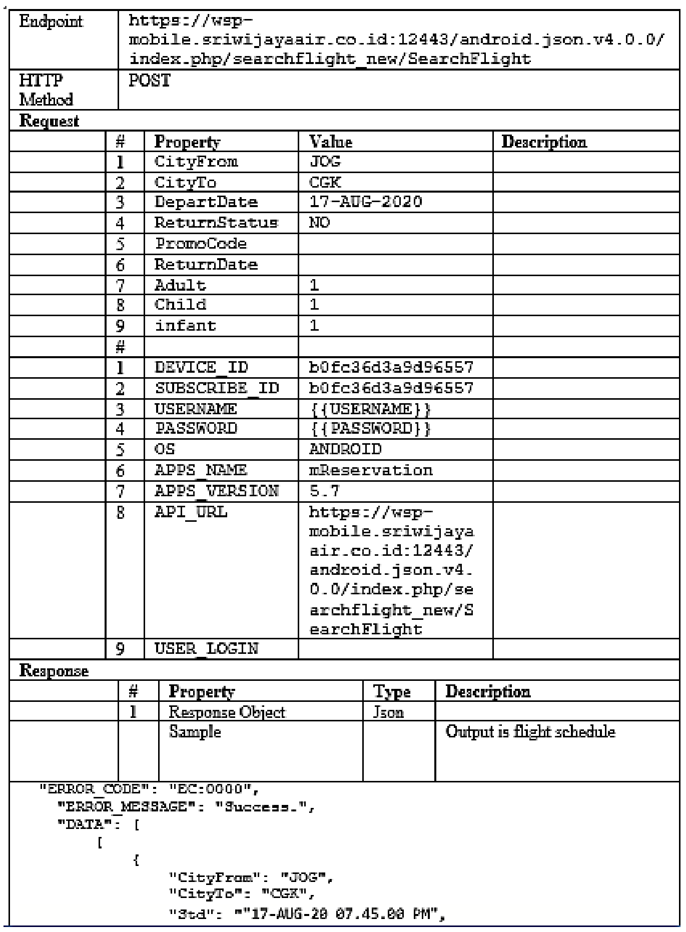
13 variabel for scenario test:

A: City from	F: Number of Passengers	K: Phone Number
T: City To	G: Personal Data of Passengers	L: Payment Method
C: Depart Date	H: Unique Key	M: Payer's Email
D: Oneway/Return	I: Search Key	
E: Return Date	J: Booking Code	

Input														Output
TeFt CaFc	A	T	C	D	E	F	G	H	I	J	K	L	M	
1	T	T	T	T	T	T	T	T	T	T	T	T	T	OK
2	T	T	T	T	F	T	T	T	T	T	T	T	T	OK
3	T	T	T	T	F	T	T	T	F	F	T	T	T	OK
4	T	T	T	T	T	F	T	T	T	T	T	T	T	OK
5	T	T	T	F	T	T	T	F	F	F	F	F	F	OK
6	T	F	F	F	T	T	T	T	T	T	T	T	T	OK
7	T	T	T	T	F	F	T	T	T	T	T	T	T	OK
8	T	T	F	T	T	T	T	T	T	T	T	T	T	OK
9	T	T	F	F	F	T	T	T	T	T	T	T	T	OK
10	T	F	F	T	T	T	T	F	F	F	F	F	F	OK
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
156	T	F	F	F	F	F	F	F	F	F	F	F	F	OK

T: True, F: False

Fig. 4 Test scenario.



Endpoint	https://wsp-mobile.sriwijayaair.co.id:12443/android.json.v4.0.0/index.php/searchflight_new/SearchFlight			
HTTP Method	POST			
Request				
	#	**Property**	**Value**	**Description**
	1	CityFrom	JOG	
	2	CityTo	CGK	
	3	DepartDate	17-AUG-2020	
	4	ReturnStatus	NO	
	5	PromoCode		
	6	ReturnDate		
	7	Adult	1	
	8	Child	1	
	9	infant	1	
	#			
	1	DEVICE_ID	b0fc36d3a9d96557	
	2	SUBSCRIBE_ID	b0fc36d3a9d96557	
	3	USERNAME	{{USERNAME}}	
	4	PASSWORD	{{PASSWORD}}	
	5	OS	ANDROID	
	6	APPS_NAME	mReservation	
	7	APPS_VERSION	5.7	
	8	API_URL	https://wsp-mobile.sriwijayaair.co.id:12443/android.json.v4.0.0/index.php/searchflight_new/SearchFlight	
	9	USER_LOGIN		
Response				
	#	**Property**	**Type**	**Description**
	1	Response Object	Json	
		Sample		Output is flight schedule

```
  "ERROR_CODE": "EC:0000",
    "ERROR_MESSAGE": "Success.",
    "DATA": [
        [
            {
                "CityFrom": "JOG",
                "CityTo": "CGK",
                "Std": ""17-AUG-20 07.45.00 PM",
```

Fig. 5 Unit test on search feature.

3) The Implementation on Payment: This stage dealt with the implementation of the Payment feature which contained Payment Request to request the payment methods through Automated Teller Machine. The function of payment response is to respond the request by displaying the payment code.

4) The Implementation on Booking Info: This stage concerned with the implementation of the Booking Info feature that contained Booking Info Request to request the filling of the payment code. The payment response is to respond the request by displaying the details of booking information.

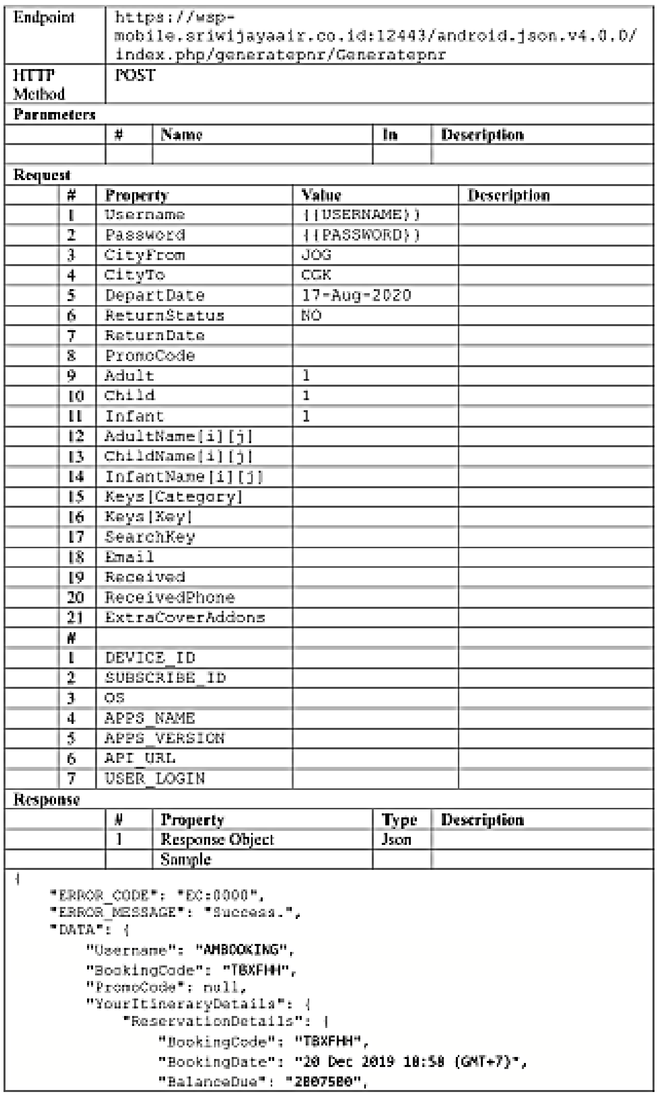
B. Integrity Testing

In this stage of integrity testing, the test was performed by testing the integrity of the microservice. The test was conducted by ensuring the sequential or the sequence of the test was maintained. The test in this stage used the metric to ensure whether or not the inputted sequence similar to the received output, for example by making a reservation then make the payment and to ensure whether the received output from the reservation came first or not.

The researchers tried to commence the stage of ticket search at first by placing the flight date on August 20th, 2020. Search Feature was conducted when a ticket search and straight payment could not be done for not having a booking code which was supposed to be given on the book feature. The error output are displayed in Fig. 6.

In the integrity test, the test for the obtained response time was carried out. According to Nielson's user interaction theory in determining the response time limit, there are three points about it which are explained as follows [12].

- 0.1 sec: Feels immediate to the user. No additional feedback needed.
- 1.0 sec: Tolerable, but it does not feel immediate. Some feedback needed.



Endpoint	https://wsp-mobile.sriwijayaair.co.id:12443/android.json.v4.0.0/index.php/generatepnr/Generatepnr
HTTP Method	POST

Parameters

	#	Name	In	Description

Request

	#	Property	Value	Description
	1	Username	{{USERNAME}}	
	2	Password	{{PASSWORD}}	
	3	CityFrom	JOG	
	4	CityTo	CGK	
	5	DepartDate	17-Aug-2020	
	6	ReturnStatus	NO	
	7	ReturnDate		
	8	PromoCode		
	9	Adult	1	
	10	Child	1	
	11	Infant	1	
	12	AdultName[i][j]		
	13	ChildName[i][j]		
	14	InfantName[i][j]		
	15	Keys[Category]		
	16	Keys[Key]		
	17	SearchKey		
	18	Email		
	19	Received		
	20	ReceivedPhone		
	21	ExtraCoverAddons		
	#			
	1	DEVICE_ID		
	2	SUBSCRIBE_ID		
	3	OS		
	4	APPS_NAME		
	5	APPS_VERSION		
	6	API_URL		
	7	USER_LOGIN		

Response

	#	Property	Type	Description
	1	Response Object	Json	
		Sample		

```
{
    "ERROR_CODE": "EC:0000",
    "ERROR_MESSAGE": "Success.",
    "DATA": {
        "Username": "AMBOOKING",
        "BookingCode": "TBXFHH",
        "PromoCode": null,
        "YourItineraryDetails": {
            "ReservationDetails": {
                "BookingCode": "TBXFHH",
                "BookingDate": "20 Dec 2019 18:58 (GMT+7)",
                "BalanceDue": "2807580",
```

Fig. 6 Integrity test on Book feature.

- 10 sec: Maximum duration to keep user's focus on the action.

If this research is examined based on test case of 1,000 requests and the response time is more than 1,000 ms, the result is shown in Table III. Based the table, Sriwijaya Air has the response time for the toleration score of the users which is more than 1,000 ms. Therefore, it needs an improvement on the API system to increase the toleration score of the users.

C. End-to-end Testing

In this stage of end-to-end testing, the test was performed by testing the start and the end of the microservice. The testing used the metric to ensure whether the starting process of the test had the expected output or not, for example was the reservation process. From the final result (output), it can be shown whether the ticket is appropriately ordered or not.

In this stage, the checking process can be done by using the booking code given from the ticket ordering process. The received booking code can be checked through Sriwijaya Air's website. If the ticket is ordered appropriately, it will display the ticket's detail information in accordance with the input alongside the ticket's status. The result can be observed in Fig. 7.

TABLE III
RESPONSE TIME TESTING

Testing Sequence	1,000 ms
2	401
3	587
4	329
5	617
6	320
7	400
8	279
9	587
10	403

D. Negative Test

This research was also conducted a negative test. This test was done using the method of black-box. The black-box test emphasizes on the system's function. This is intended to make sure that the software runs smoothly and to minimize the errors. One of the most important things before a software application is used by a client is to have an error reporting facility to send known error or bug to the developer for action then it becomes a bug free application as the result.

1) Negative Test on Search Feature: This negative test was conducted by not filling the input completely on the API's structure or by leaving one of the inputs blank. For example is in the Search feature, the passengers are required to input the departure city, destination city, dates, passenger's name, etc., but then one of them is left blank. If the form is left blank, there will be no displayed data or it will lead to an error.

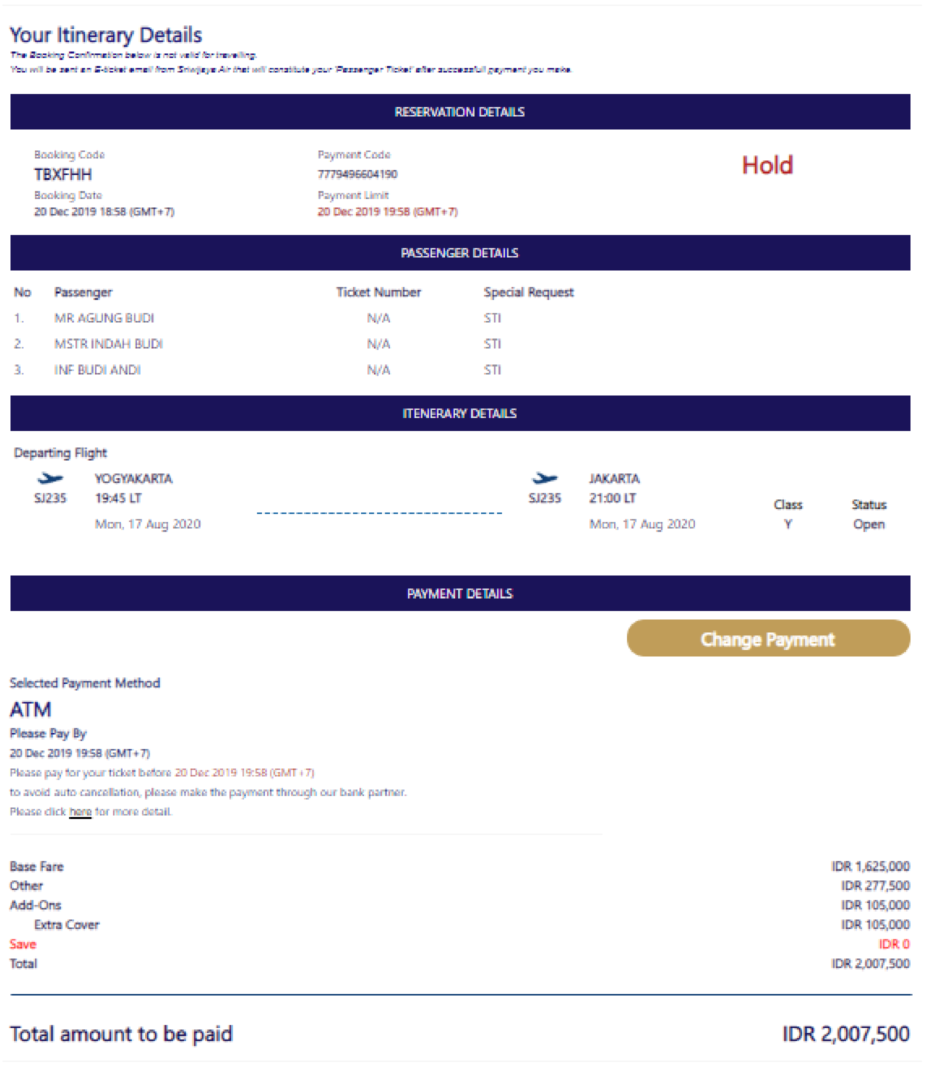
2) Negative Test on Book Feature: This negative test was performed by not filling the input completely on the API's structure or by leaving one of the inputs blank. For example is when the passengers skip the Search feature that contains the input for departure city, destination city, dates, passenger's name, etc. If they are left blank, there will be no displayed data or it will lead to an error. Such response can be seen in Fig. 8.

3) Negative Test on Payment Feature: This negative test was conducted by placing the incorrect booking code on the API's stucture. If it is given a wrong input, the system will respond by not displaying the data or it will lead to an error.

4) Negative Test on Booking Info Feature: This negative test was performed by placing the incorrect booking code on the API's stucture. If it is given a wrong input, the system will respond by not displaying the data or error.

V. ANALYSIS

According to the previous test result, three testing stages which are being proposed can be applied by Sriwijaya Air. The three testing stages are unit testing, integrity testing, and end-to-end testing. Besides, the test result also shows that there is a need to conduct an improvement on the API such as increasing the speed of the API to get the higher score for the users'



Your Itinerary Details

The Booking Confirmation below is not valid for travelling.

You will be sent an E-ticket email from Sriwijaya Air that will constitute your 'Passenger Ticket' after successfull payment you make.

RESERVATION DETAILS

Booking Code	Payment Code	Hold
TBXFHH	7779496604190	
Booking Date	Payment Limit	
20 Dec 2019 18:58 (GMT+7)	20 Dec 2019 19:58 (GMT+7)	

PASSENGER DETAILS

No	Passenger	Ticket Number	Special Request
1.	MR AGUNG BUDI	N/A	STI
2.	MSTR INDAH BUDI	N/A	STI
3.	INF BUDI ANDI	N/A	STI

ITENERARY DETAILS

Departing Flight

SJ235	YOGYAKARTA 19:45 LT Mon, 17 Aug 2020	SJ235	JAKARTA 21:00 LT Mon, 17 Aug 2020	Class Y	Status Open

PAYMENT DETAILS

Change Payment

Selected Payment Method

ATM

Please Pay By

20 Dec 2019 19:58 (GMT+7)

Please pay for your ticket before 20 Dec 2019 19:58 (GMT+7)

to avoid auto cancellation, please make the payment through our bank partner.

Please click here for more detail.

Base Fare	IDR 1,625,000
Other	IDR 277,500
Add-Ons	IDR 105,000
Extra Cover	IDR 105,000
Save	IDR 0
Total	IDR 2,007,500

Total amount to be paid IDR 2,007,500

Fig. 7 End-to-end testing.

toleration score. The toleration score means the score of the response time that the users have when they interact with the system and the length of the time has to be tolerable enough for the users. It means, if the length of the time is more than the predetermined time which becomes the standard, then the users will be considered to give no toleration for it.

The testing stages that consist of unit testing, integrity testing, and end-to-end testing are applicable to Sriwijaya Air's API open based system because this system is still under development. The unit testing on the airline was based on the smallest features. In this research, the researchers took four features which were search, book, payment, and booking info. The integrity testing on the airline was based on the connections between one feature and another feature. Besides, it was also based on the system's sequential process. The end-to-end testing on the airline's system was tested based on the results after conducting testing stages.

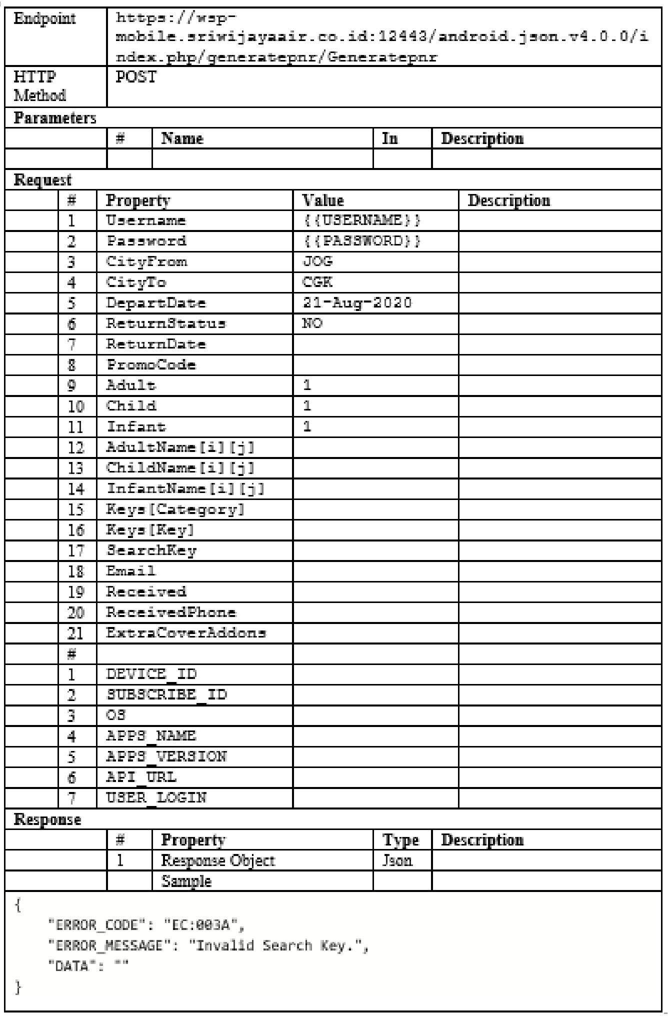
Several airlines have the procedure to book the ticket online. Some of those airlines are Sriwijaya Air, Batik Air, and Citilink. The three proposed testing stages which are discussed before can be applied on those airlines. However, the airline's API has to be able to be accessed publicly. The API structure has to be able to be seen from the ticket booking process and data structure which is used during the ticket booking process. This data structure is possible to be seen from the inputted field during the online ticket booking process. It is because the API structure that is used in the back-end is not really different from the inputted field in the front-end. The procedure to book a ticket online in each airlines can be found in Table IV.

Table IV shows that those modules are the modules that have the possibility to be the API structure of Batik Air and Citilink. For example, the flight search module of the three airlines have the input which can be observed in Table V.

From the input in Table IV, Batik Air, and Citilink's flights have the same input with Sriwijaya Air. Therefore, the three testing stages can be applied in Batik Air and Citilink.

The systems other than in Sriwijaya Air may have these testing stages with the following conditions.

1. Several processes involving ticket reservation have some similarities and some differences, such as the API's structure of each airline that is different to each other even though it has similar parameters.



Endpoint	`https://wsp-mobile.sriwijayaair.co.id:12443/android.json.v4.0.0/index.php/generatepnr/Generatepnr`
HTTP Method	POST

Parameters

	#	Name	In	Description

Request

	#	Property	Value	Description
	1	Username	{{USERNAME}}	
	2	Password	{{PASSWORD}}	
	3	CityFrom	JOG	
	4	CityTo	CGK	
	5	DepartDate	21-Aug-2020	
	6	ReturnStatus	NO	
	7	ReturnDate		
	8	PromoCode		
	9	Adult	1	
	10	Child	1	
	11	Infant	1	
	12	AdultName[i][j]		
	13	ChildName[i][j]		
	14	InfantName[i][j]		
	15	Keys[Category]		
	16	Keys[Key]		
	17	SearchKey		
	18	Email		
	19	Received		
	20	ReceivedPhone		
	21	ExtraCoverAddons		
	#			
	1	DEVICE_ID		
	2	SUBSCRIBE_ID		
	3	OS		
	4	APPS_NAME		
	5	APPS_VERSION		
	6	API_URL		
	7	USER_LOGIN		

Response

	#	Property	Type	Description
	1	Response Object	Json	
		Sample		

```
{
    "ERROR_CODE": "EC:003A",
    "ERROR_MESSAGE": "Invalid Search Key.",
    "DATA": ""
}
```

Fig. 8 Negative test on Book feature.

2. The problem is that several APIs are private. Therefore, it may be difficult to get the legal API. It is different with an open API that can be accessed publicly and simply by signing in as a user.

VI. Conclusion

The purpose of this research is to discuss the application of the integrity testing between microservices as one of the microservice testing stages. This research proposes the microservice testing stages for the case study which is Sriwijaya Air, one of the airline in Indonesia.

Sriwijaya Air was tested using the three testing stages. This research applied black-box testing strategy and applied the positive and negative testing as well.

The testing was done manually by testing the four features before booking the ticket. Those four features are search, book, payment, and booking info. Unit testing was done to test the functions every module, integrity test was applied to test the relationship between the microservices, and end-to-end testing was conducted to test the result after applying the previous tests. The three testing stages are sufficient to represent other testing stages. It is because they tested the smallest functions individually, the service between microservices, and the overall system result. The results of the Sriwijaya Air's API microservice test are explaind as follows.

1. The result of the three tests shows that the microservices are related to one and another. It means, if one microservice is not used, the system process cannot be used until the end and the result of the microservice will be used in another microservice. Therefore, the sequential of the system stages is maintained.
2. The obtained response time is not tolerable enough for the users. It needs an improvement in Sriwijaya Air's API in order to increase the speed of its API so that the toleration score of the response time is also increased.
3. The three testing stages can be applied by other airlines as long as the API of the airlines can be accessed publicly.
4. These three stages are not only easy to do and have short estimation of time, but also have the low cost which is important.

It can be concluded that integrity testing which is applied in Sriwijaya Air's API microservice test is able to test the integrity between microservices. Integrity testing can be combined with the unit testing and end-to-end testing, so that it generates short time estimation and low cost. The three stages are also possible to be used by the developer to test the microservice.

TABLE IV
THE COMPARISONS OF AIRLINES' MODULE

No	Sriwijaya Air	Batik Air	Citilink
1	Flight Search	Flight Search	Flight Search
2	Passengers' Data Filling	Passengers' Data Filling	Passengers' Data Filling
3	Payment Data Filling	Baggage Choosing	Seat Selecting
4	Getting the Booking Code	Payment Data Filling	Payment Data Filling
5		Getting the Booking Code	Getting the Booking Code

TABLE V
THE INPUT IN FLIGHT SEARCH MODULE

No	Sriwijaya Air	Batik Air	Citilink
1	City From	City From	City From
2	City To	City To	City To
3	Depart Date	Depart Date	Depart Date
4	Return Status	Return Status	Return Status
5	Promo Code	Return Date	Return Date
6	Return Date	Adult	Adult
7	Adult	Child	Child
8	Child	Infant	Infant
9	Infant		Currency
10			Flight Class

References

[1] M. Rahman and J. Gao, "A Reusable Automated Acceptance Testing Architecture for Microservices in Behavior-driven Development," *Proc. - 9th IEEE Int. Symp. Serv. Syst. Eng. IEEE SOSE 2015*, 2015, pp. 321–325.

[2] V. Heorhiadi, S. Rajagopalan, H. Jamjoom, M. K. Reiter, and V. Sekar, "Gremlin: Systematic Resilience Testing of Microservices," *Proc. - Int. Conf. Distrib. Comput. Syst.*, 2016, pp. 57–66.

[3] C. Richardson (2019) "What are Microservices," [Online],

http://microservices.io/index.html, access date: 10-Jan-2020.

[4] C. Carneiro Jr and T. Schmelmer, *Microservices From Day One: Build Robust and Scalable Software from the Start*, New Yori, USA: Apress, 2016.

[5] S.P. Ma, C.Y. Fan, Y. Chuang, W.T. Lee, S.J. Lee, and N.L. Hsueh, "Using Service Dependency Graph to Analyze and Test Microservices," *Proc. - Int. Comput. Softw. Appl. Conf.*, 2018, pp. 81–86.

[6] J.P. Sotomayor, S.C. Allala, P. Alt, J. Phillips, T.M. King, and P.J. Clarke, "Comparison of Runtime Testing Tools for Microservices," *Proc. - Int. Comput. Softw. Appl. Conf.*, 2019, pp. 356–361.

[7] M.J. Kargar and A. Hanifizade, "Automation of Regression Test in Microservice Architecture," *2018 4th Int. Conf. Web Res. ICWR 2018*, 2018, pp. 133–137.

[8] (2020) "Sriwijaya Air," [Online], https://www.sriwijayaair.co.id/, access date: 8-Feb-2020.

[9] G.J. Myers, C. Sandler, and T. Badgett, *The Art of Software Testing*, 3rd ed., Hoboken, USA: John Wiley & Sons, Inc., 2012.

[10] A. Ghahrai (2017) "Microservice Testing," [Online], https://devqa.io/qa/testing-microservices-beginners-guide, access date: 15-Mar-2020.

[11] C. Richardson (2019) "How to Test a Microservice." [Online], https://microservices.io/testing/ access date: 15-Mar-2020.

[12] J. Nielsen, *Usability Engineering*, Cambridge: USA: Academic Press, 1993.