RANCANG BANGUN SISTEM SINKRONISASI DATA MENGGUNAKAN *GOOGLE CLOUD PUB/SUB* DAN *FLASK* DI PT XYZ

George Schumy[1], Yeremia Alfa Susetyo[2]
[1 2] Teknik Informatika S1 Universitas Kristen Satya Wacana
672018014@student.uksw.edu

ABSTRAK

Perkembangan dunia teknologi informasi yang semakin berkembang tidak lepas dari penggunaan *Cloud Computing*. Dengan semakin berkembangannya bisnis PT XYZ, penggunaan media pengiriman dan penyimpanan data menjadi prioritas utama untuk dikembangkan. Dengan menggunakan *Cloud Computing*, pengguna dapat menyimpan dan mengakses data menggunakan internet. Namun, menjadi masalah jika dalam melakukan sinkronisasi data *local* ke *Cloud* bisa terjadi ketidakstabilan koneksi yang menyebabkan kegagalan pada sinkronisasi data ke *Cloud*. Sehingga pada penelitian ini dibangunlah sistem yang dapat menjaga sinkronisasi data menggunakan teknologi *Publish Subscribe* yang ada pada *Google Cloud Pub/Sub* dan *Flask* sebagai penyedia *RESTful* API dengan menggunakan bahasa pemrograman *python*. Dengan menggunakan *Google Cloud Pub/Sub*, pengiriman data akan dilakukan sekali dan sistem akan menyalin data ke beberapa zona untuk memastikan data/pesan tersampaikan. Selain itu semua data akan dienkripsi dan dilindungi. Pada penelitian ini dilakukan rancang bangun menggunakan metode *Research and Development*. Berdasarkan hasil penelitian sistem dinilai mampu melakukan sinkronisasi data *offline* ke *cloud* dan menjaga keabsahan data di *CloudSQL*. Selain itu fitur *Dead Letter Queue* dapat membantu *Backend Developer* PT XYZ dalam melihat dan memperbaiki data. Kesimpulan tersebut dapat di dukung dari hasil kuesioner yang memperoleh persentase sebesar 88% dan digolongkan "Sangat Setuju" terhadap rancangan sistem yang dibuat.

Keyword : *Google Cloud Pub/Sub, Flask, Python, Research and Development.*

1. PENDAHULUAN

Perkembangan dunia teknologi informasi saat ini sangat cepat sehingga kini banyak perusahaan swasta maupun pemerintah berusaha meningkatkan usahanya terutama dalam bidang bisnis yang berkaitan dengan teknologi informasi. Penggunaan teknologi *Cloud Computing* dapat menjadi solusi pengehematan biaya dalam pengembangan usaha dengan menggunakan infrastruktur teknologi informasi [1]. Pada tahun 2018 *Global Cloud Computing Scorecard* menyatakan bahwa pengembangan *Cloud Computing* di Indonesia masih tergolong rendah. Indonesia menempati posisi ke-23 dari 24 ekonomi IT terkemuka [2].

Dengan semakin berkembangnya bisnis, menyebabkan penggunaan media pengiriman dan penyimpanan data menjadi prioritas utama PT.XYZ untuk mengembangkannya lebih jauh. Dengan menggunakan *Cloud Computing* pengguna dapat menyimpan semua data di *Cloud* dan menggunakan koneksi internet untuk mengaksesnya. Namun akan menjadi masalah jika pengguna berada di daerah terpencil sehingga kesulitan mendapatkan koneksi internet atau sedang dalam kondisi *Offline* sehingga Pengguna tidak dapat mengakses database di *Cloud* sehingga perlu dibuatnya database *local* yang disinkronisasikan dengan *Cloud* [3]. Dalam melakukan sinkronisasi data *local* ke *Cloud* jika kondisi jaringan sedang tidak stabil, dapat menyebabkan kegagalan pada sinkronisasi data ke *Cloud*. Kegagalan ini dapat terjadi karena partisi dalam jaringan, kehilangan atau kerusakan paket, kemacetan, kegagalan node atau tautan tujuan, dll [4].

Oleh karena itu perlu adanya sistem yang dapat menjaga sinkronisasi data ke *Cloud* sehingga dapat mengurangi kegagalan yang dapat diakibatkan oleh data yang bermasalah maupun koneksi yang kurang stabil dalam melakukan sinkronisasi data. *Publish-subscribe* adalah arsitektur yang dapat menangani pengiriman data yang di *publish* oleh *publisher* ke beberapa *subscriber* atau pengguna berdasarkan topik yang dipilihnya melalui broker sebagai perantaranya [5]. *Google Cloud Pub/Sub* merupakan salah satu broker atau penyedia layanan *Publish-Subscribe*. Dengan menggunakan sistem *Google Cloud Pub/Sub*, pengiriman data akan dilakukan sekali dan sistem akan menyalin data ke beberapa zona untuk memastikan data/pesan tersampaikan. Selain itu semua data akan dienkripsi dan dilindungi [6]. Dari keunggulan yang ditawarkan, *Google Cloud Pub/Sub* menjadi pilihan yang baik dalam melakukan sinkronisasi data *Offline* ke *Cloud*.

Berdasarkan latar belakang yang telah disebutkan, penelitian ini bertujuan untuk merancang sistem sinkronisasi data *Offline* ke *Cloud* agar dapat memenuhi kebutuhan PT XYZ dalam menjaga sinkronisasi data ke *Cloud*. Penelitian ini akan menghasilkan sebuah rancangan sistem sinkronisasi data *Offline* ke *Cloud database* menggunakan *Google Cloud Pub/Sub* sehingga dapat menjamin data dapat tersinkronisasi dengan baik serta dapat memperbaiki data yang bermasalah sehingga tidak dapat tersampaikan.

2. TINJAUAN PUSTAKA

Penelitian yang berjudul "Optimasi Kinerja *Point Of Sale* (POS) Dengan Penerapan Sinkronisasi Database Menggunakan Middleware". Pada penelitian ini menjelaskan mengenai pembuatan algoritma sinkronisasi menggunakan *middleware* pada aplikasi *Point Of Sale* (POS) pada toko butik bernama RAIN. Kasus pada penelitian ini adalah infrastruktur jaringan yang kurang memadai untuk penerapan database terpusat sehingga kegiatan jual beli pada toko terganggu. Selain itu, sinkronisasi database pada toko butik RAIN masih dilakukan secara manual melalui email dari setiap cabang butik ke kantor pusat. Email dikirimkan pada setiap akhir jam kerja. Setelah data diterima kantor pusat maka kantor pusat akan memasukkan data ke database pusat dan apabila terdapat perubahan data, maka kantor pusat akan mengekspor data ke setiap email cabang butik untuk melakukan update pada databasenya [7]. Pada penelitian ini menghasilkan sinkronisasi otomatis sehingga tidak perlu dilakukan secara manual. Dan ketika jaringan sedang *offline* maka pengiriman data akan ditunda hingga jaringan kembali *online*. Tetapi pada penelitian ini tidak menjelaskan permasalahan jika jaringan yang tidak stabil ketika melakukan sinkronisasi data apakah data diterima secara tepat atau terdapat data yang rusak atau tidak tersinkronisasi dengan baik.

Penelitian yang berjudul "Monitoring Penggunaan Daya Listrik Menggunakan Protokol MQTT berbasis Web". Pada penelitian ini menjelaskan tentang pembuatan sistem monitoring pemakaian daya listrik menggunakan *realtime* protokol MQTT berbasis *Publish/Subscribe*. Pembuatan sistem perlu dilakukan karena pendistribusian data pemakaian listrik masih dilakukan secara konvensional dan pengambilan data dengan waktu yang masih tidak menentu [8]. Pada penelitian dijelaskan bahwa penggunaan Protokol MQTT berbasis *Publish/Subsrcribe* diterapkan karena dapat berjalan dengan kondisi bandwidth yang rendah serta *latency* yang tinggi.

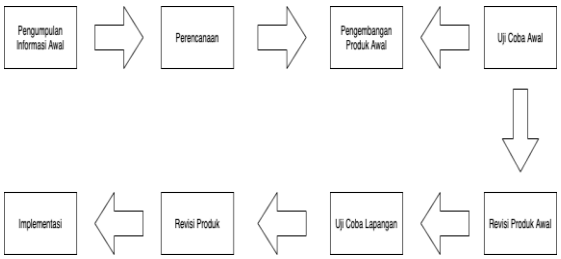
Penelitian yang berjudul "Implementasi Sistem Tracking Posisi Ambulans pada Smart Dispatcher Menggunakan Metode Komunikasi Publish/Subscribe". Pada penelitian ini penulis menjelaskan tentang penggunaan metode *Publish/Subscribe* dengan protokol MQTT pada pengiriman data ambulans yang berupa lokasi dan status ambulans ke pusat penerima data. Data tersebut berguna untuk menyimpan lokasi ambulans dan melakukan perhitungan jarak terdekat terhadap ambulans-ambulans dan pasien [9]. Pada penelitian ini penggunaan teknologi *Publish/Subscribe* lebih memfokuskan kecepatan pengiriman data dengan menggunakan teknologi *Publish/Subscribe* tanpa memperhatikan segi pengiriman data apakah terjadi *Network Fault*.

Penelitian yang berjudul "Implementasi Perangkat *Mobile Publisher Subscriber* Sebagai Perantara Pengiriman Data Sensor Dari Lapangan Kepusat Data". Pada penelitian ini menjelaskan mengenai pengembangan aplikasi berbasis android yang digunakan untuk mengambil data sensor di lapangan. Sistem mengimplementasikan komunikasi *publish/subscribe* sebagai perantara pengiriman data sensor di lapangan kepusat data dengan menggunakan protokol MQTT [10]. Pada penelitian ini menghasilkan sistem yang dapat melakukan pengiriman data sensor dengan komunikasi *publish/subscribe* menggunakan protokol MQTT.

Penelitian yang berjudul "Implementasi Paradigma Publish-Subscribe Untuk Menjalankan Event-Based Monitoring Pada Sistem Pengamatan Kandang Ternak". Pada penelitian ini menjelaskan tentang pembuatan sistem untuk mengatasi permasalahan jeda waktu pemeriksaan kandang. Penelitian ini menggunakan protokol MQTT karena memiliki kemampuan yang handal,ringan, dan arsitektur yang sederhana sehingga mudah di implementasikan. Pada penelitian ini proses pengiriman data dilakukan jika terjadi kondisi eksterem. Kondisi ekstrem ditentukan dengan memberikan batas pada nilai suhu udara pada kandang ayam [11].

3. METODE PENELITIAN

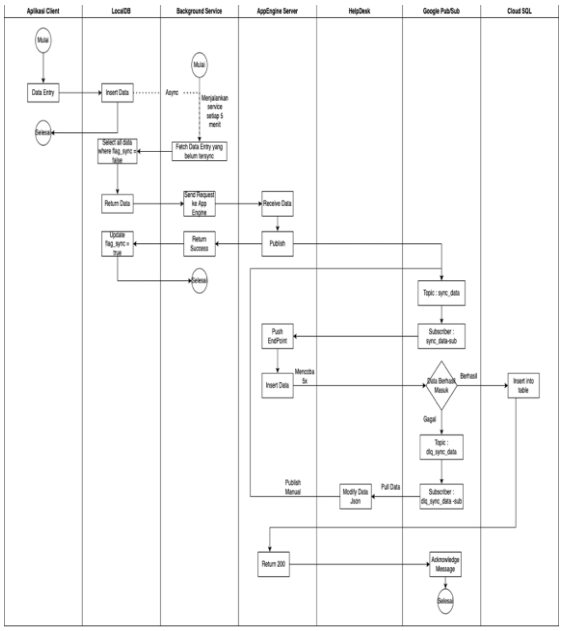
Model pengembangan yang dipakai dalam penelitian ini merupakan model pengembangan yang dikemukakan oleh Sugiyono yaitu *Research and Development* (R&D) [12] yang kemudian disesuaikan dengan tujuan dan kondisi penelitian yang sebenarnya. Bentuk penelitian ini dapat dilihat pada Gambar 1.



Gambar 1. Model *Research and Development*

Pada Gambar 1 menjelaskan setiap proses yang dilakukan dalam penelitian, yang pertama pengumpulan informasi awal. Pengumpulan informasi awal dilakukan untuk mendapatkan permasalahan PT XYZ yaitu ingin melakukan sinkronisasi data *offline* ke *Cloud* dengan menggunakan sistem yang dapat memastikan bahwa data tersinkronisasi ke *Cloud*. Kemudian setelah mendapatkan informasi mengenai permasalahan maka dilakukan perencanaan dan menghasilkan perancangan sistem *Google Cloud Pub/Sub* dan *Flask* untuk melakukan sinkronisasi data *offline* ke *Cloud* menggunakan bahasa pemrograman *python*. Adapun manfaat dari menggunakan *Google Cloud Pub/Sub* adalah dapat melakukan sinkronisasi data tanpa mengkhawatirkan koneksi yang tidak stabil dan melakukan sinkronisasi data kembali jika data

tidak sampai ke *endpoint subscriber* yang merupakan *CloudSQL*. Activity sistem dapat dilihat pada Gambar 2



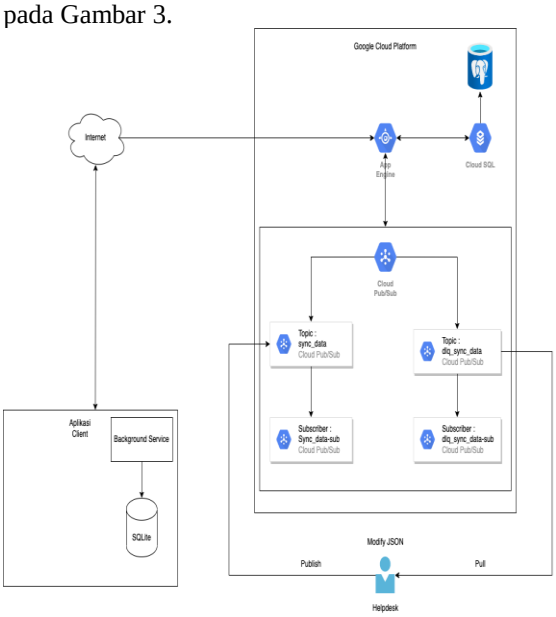
Gambar 2. Activity sistem sinkronisasi data menggunakan *Google Cloud Pub/Sub* dan *Flask* (Sumber: Dokumen Pribadi)

Setelah melakukan perencanaan maka dilakukan pengembangan produk awal. Pembangunan sinkronisasi data dilakukan dengan menggunakan *App Engine*,*Google Pub/Sub*,Python,SQLite,Flask dan PostgreeSQL yang terdapat pada *CloudSQL*. Setelah melakukan pengembangan kemudian dilakukan uji coba awal di lapangan dengan melibatkan 5 orang pegawai (*Infromation Technology*) IT pada PT. XYZ. Dari hasil uji-coba kemudian dilakukan diskusi mengenai produk tersebut bersama dengan team IT untuk mengetahui permasalahan pada sistem. Berdasarkan permasalahan uji coba awal selanjutnya dilakukan revisi produk awal kemudian dilakukan kembali uji coba lapangan yang melibatkan 20 *Backend Developer* PT XYZ selaku pengguna sistem. Data yang dikumpulkan berupa diskusi dan kuesioner. Setelah uji coba lapangan kemudian dilakukan revisi produk jika ada permasalahan pada uji coba lapangan berdasarkan hasil diskusi dan kuesioner. Pada tahap akhir dilakukan implementasi sistem ke dalam produk aplikasi PT.XYZ dengan menyesuaikan banyaknya topic dan subscriber berdasarkan tabel yang ingin disinkronisasikan.

4. HASIL DAN PEMBAHASAN

Penelitian ini menghasilkan rancangan sistem yang dapat digunakan dalam melakukan sinkronisasi data *offline* ke *cloud* dan dibangun dengan menggunakan teknologi *Google Cloud Pub/Sub* dan *Flask* yang terdapat pada *library python*. Dalam pembuatan sistem, diperlukan database *offline* dan *online* untuk melakukan sinkronisasi data.

Pembuatan database *offline* dibuat dengan menggunakan SQLite sedangkan untuk database *online* dibuat dengan menggunakan *Postgre*SQL yang ada pada *Google CloudSQL* Sistem juga dibuat dan di *deploy* ke dalam *Google App Engine* yang merupakan platform untuk mengembangkan dan menghosting aplikasi web [13] sehingga dapat terhubung dengan *Google Cloud Pub/Sub* dan *Google CloudSQL*. Arsitektur sistem dapat dilihat pada Gambar 3.



Gambar 3. Arsitektur Sistem Sinkronisasi Data (Sumber: Dokumen Pribadi)

Setelah merancang arsitektur sistem, kemudian dilanjutkan dengan pembuatan *topic* dan *subscriber* pada *Google Cloud Pub/Sub* agar dapat terhubung dengan sistem yang dibangun.

Tabel 1. Nama *Topic* dan *Subscriber*

Topic	Subscriber	Tipe	Keterangan
sync_data	sync_data-sub	Push	Proses Sinkronisasi Data
dlq_sync_data	dlq_sync_data-sub	Pull	*Dead Lettering* Sinkronisasi Data

Pada Table 1 merupakan nama topic dan subscriber yang dipakai untuk melakukan sinkronisasi data. *Topic* merupakan tempat untuk *publisher* mengirimkan data. Pada penelitian ini terdapat 2 topic yaitu sync_data untuk menerima data yang berhasil dikirimkan ke *CloudSQL* dan dlq_sync_data untuk menerima pesan yang gagal akibat ketidaksesuaian bentuk data dengan table pada *CloudSQL*. Selain topic juga terdapat subscriber yang berguna untuk menerima aliran data dari topic ke aplikasi yang dimana pada penelitian diarahkan kepada *Google CloudSQL*. Terdapat 2 subscriber yang digunakan yaitu sync_data-sub yang menggunakan metode *push endpoint* dalam mengirimkan data ke *Google CloudSQL* dan

dlq_sync_data-sub yang menggunakan metode *pull* untuk mendapatkan data yang tidak tersinkronisasi akibat kesalahan bentuk data.

Kode Program 1. Kode program untuk *publish* dan *receive* data

Kode Program

```
import base64
import json
from flask import Blueprint, request, jsonify
from controllers.sync.model import Sync
from globals.pubsub import PubSub

sync_endpoints = Blueprint("sync", __name__)

@sync_endpoints.route("/set",
methods=["POST"])
def sync_data():
    data_request = request.get_json()
    try:
        pub_sub = PubSub(topic_path="sync_data",
                         project_id='hippa-66637')
        pub_sub.publish(data_request)
        status, message, code = True, "Data send to
pubsub", 200
    except Exception as e:
        status, message, code = False, repr(e), 400
    return jsonify({'status': status, 'message':
message}), code

@sync_endpoints.route('/receive',
methods=["POST"])
def receive_sync_pubsub():
    message = json.loads(request.data.decode('utf-
8'))
    print("Message pubsub : ", message)
    content =
base64.b64decode(message['message']['data'])
    data_request = json.loads(content.decode('utf-
8'))
    print("Data pubsub : ", message)
    model = Sync()
    status, result = model.sync_data(data_request)
    return jsonify(result), status
```

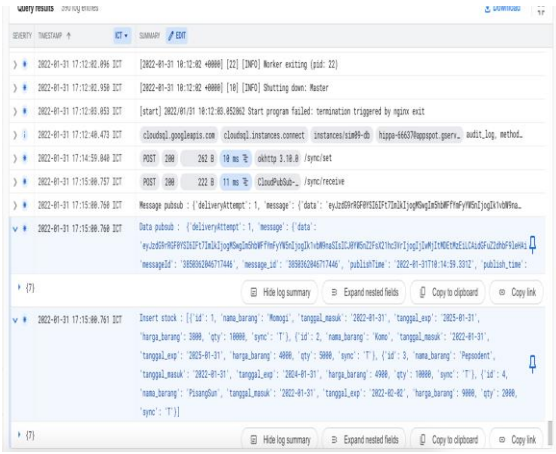
Kode program 1 merupakan fungsi yang digunakan untuk melakukan *push* ke *topic*. Pada baris 1-5 merupakan library / package yang digunakan untuk melakukan sinkronisasi data. Baris 7-9 merupakan routing yang digunakan untuk mengakses fungsi sync_data. Pada baris 11 digunakan untuk mendapatkan data yang berupa json. Setelah itu pada baris 13-17 merupakan fungsi untuk memberikan *topic* dan project_id yang ada pada *Google Cloud Platform* ketika *topic* dan project_id benar maka akan memberikan return 200. Pada baris 19-21 merupakan return error 400 jika *topic* atau project_id tidak sesuai. Sama seperti baris 9 baris 23 merupakan routing yang digunakan untuk mengakses fungsi receive_sync_pubsub. Pada baris 25-26 digunakan untuk mendapatkan data json yang dienkripsi pada saat melakukan publish. Pada baris 27-29 dilakukan dekripsi data json untuk mendapatkan bentuk json yang sebenarnya. Pada baris 30-32 merupakan object class untuk melakukan insert data ke *CloudSQL* dan memberikan return berupa json jika sukses ataupun gagal.

Kode Program 2. Kode program untuk insert data ke CloudSQL

Kode Program

```
from globals.db import DatabaseServer

class Sync:
    def __init__(self):
        self._db = DatabaseServer()

    def sync_data(self, param):
        conn = self._db.get_connection()
        batch_responses = dict()
        is_insert_success = False

        stokData = param.get('stokData')
        if stokData is not None:
            print("Insert stock :", stokData)
            status, message =
self.insert_stock_data(conn, stokData)
            batch_responses['stockDatas'] = {
                'status': status,
                'message': message
            }
            is_insert_success = status

        if is_insert_success:
            conn.commit()
            self._db.close_connection(conn)
            return 200, batch_responses
        else:
            self._db.close_connection(conn)
            return 400, {
                'syncStatus': batch_responses,
                'syncData:': param
            }

    def insert_stock_data(self, conn, data):
        try:
     self._db.execute_many_commit_later(conn,
"""
            INSERT INTO ms_stok(
                id, nama_barang, tanggal_masuk,
tanggal_exp, harga_barang, qty, sync)
            VALUES
                (%(id)s, %(nama_barang)s,
%(tanggal_masuk)s, %(tanggal_exp)s,
%(harga_barang)s, %(qty)s, %(sync)s)
            """, data)
            return True, "Data inserted successfully."
        except Exception as e:
            conn.rollback()
            import traceback
            print(traceback.format_exc())
            return False, repr(e)
```

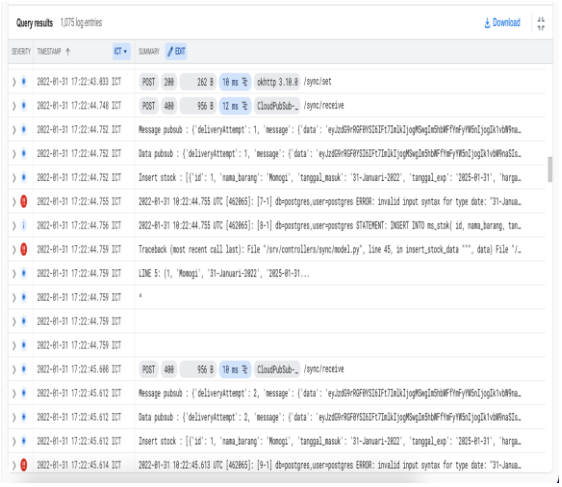
Kode Program 2 merupakan class model untuk melakukan insert data ke *Google CloudSQL*. Baris 1 merupakan package untuk membuat koneksi ke *Google CloudSQL*. Pada baris 4-5 merupakan inisialisasi class sync untuk memanggil class DatabaseServer. Pada baris 8 -11 merupakan inisialisasi untuk mendapatkan koneksi database, bentuk response secara dictionary, dan status insert. Pada baris 12-21 merupakan pengambilan data dari param setelah itu dilakukan pengecekan apakah terdapat data sehingga dapat di lakukan insert data dan hasil dari insert akan memberikan status dan messages. Pada baris 23 -32 merupakan return 200 jika sukses atau 400 jika gagal. Pada baris 34 – 50 merupakan fungsi untuk melakukan insert data dan memberikan return sukses jika berhasil atau return false jika gagal.



Gambar 4. Hasil log sukses sinkronisasi data

Gambar 4 merupakan hasil log sukses saat melakukan sinkronisasi data. Berdasarkan hasil menunjukkan bahwa ketika aplikasi melakukan push endpoint "/sync/set" memberikan return 200 yang artinya data berhasil dikirimkan ke topic. Setelah data dikirimkan ke topic data di alirkan ke subscriber yaitu "/sync/recevive" dari hasil tersebut juga didapatkan return 200 yang artinya data sudah berhasil di insert ke *CloudSQL*. Dari hasil log sukses juga menunjukkan alur data yang di enkripsi dan bentuk data setelah di dekripsi untuk melakukan insert ke dalam *Cloud* database.

Gambar 5 merupakan hasil log gagal ketika melakukan sinkronisasi data. Berdasarkan hasil menunjukkan bahwa ketika aplikasi melakukan push *endpoint* "/sync/set" memberikan hasil return 200 tetapi ketika data dialirkan ke subscriber untuk melakukan insert data ke *CloudSQL* mendapatkan return 400 dikarenakan bentuk data yang tidak sesuai dengan kolom table. Dari hasil juga dapat dilihat ketika gagal melakukan insert data yang gagal dapat dilakukan pengiriman otomatis sebanyak 5 kali.



Gambar 5. Hasil log gagal sinkronisasi data

Untuk memastikan sistem berjalan dengan semestinya, maka dilakukan pengujian sebelum sistem diimplementasikan ke aplikasi yang dimiliki PT.XYZ. Pengujian dilakukan dengan tujuan agar sistem berjalan dengan seharusnya. Peneliti melakukan pengujian dengan menggunakan metode *Black Box Testing*. *Black Box Testing* merupakan pengujian yang berfokus kepada spesifikasi fungsional dari perangkat lunak [14].

Tabel 2**.** Pengujian Sistem Sinkronisasi Data

Kasus dan Hasil Uji (Data Benar)			
Pengujian	**Yang Diharapkan**	**Pengamatan**	**Kesimpulan**
Input data sqlite sesuai dengan tipe data kolom CloudSQL	Data masuk kedalam PostgreSQL yang ada pada CloudSQL	Data masuk kedalam PostgreSQL yang ada pada CloudSQL	(V) Diterima () Ditolak
Kasus dan Hasil Uji (Data Salah)			
Input data sqlite tidak sesuai dengan tipe data kolom CloudSQL	Data masuk kedalam dlq_sync_data-sub	Data masuk kedalam dlq_sync_data-sub	(V) Diterima () Ditolak

Tabel 2 merupakan hasil dari pengujian *Black Box Testing* dari hasil uji menunjukkan bahwa data dapat masuk kedalam CloudSQL jika tipe/format data *offline* sesuai dengan table pada PostgreSQL yang terdapat pada *CloudSQL*. Ketika data tidak sesuai dengan table maka data masuk kedalam *dead lettering messages* pada dlq_sync_data-sub.

Setelah melakukan pengujian *Black Box* dilakukan uji kelayakan sistem dengan melibatkan 20 *backend developer* PT XYZ selaku pengguna sistem yang sudah mencakup lebih dari 90% dari populasi *backend developer* di PT XYZ. Pertanyaan Kuesioner dibentuk berdasarkan tujuan dari penelitian yang berfokus kepada menjaga sinkronisasi data *offline* ke *cloud* menggunakan *Google Cloud Pub/Sub* dan *Flask* serta fitur *Dead Letter Queue* yang ada pada *Google Cloud Pub/Sub*

dalam melihat *eror* dan memperbaiki data yang bermasalah.

Tabel 3**.** Perhitungan skala likert

No	Pertanyaan	Jawaban				
		STS	TS	C	S	SS
1	Apakah Pubsub dapat menyelesaikan masalah PT XYZ dalam melakukan sinkronisasi data offline ke cloud?	-	-	-	10	10
2	Apakah data yang di sinkronisasikan oleh pubsub dapat terhubung dengan baik ke cloudsql?	-	-	1	7	12
3	Apakah fitur DLQ (*Dead letter queue*) dapat membantu user dalam melihat error?	-	-	1	9	10
4	Apakah fitur DLQ (*Dead Letter queue*) dapat membantu user untuk memperbaiki data?	-	-	2	11	7

Setelah dilakukannya pengambilan dan penggabungan hasil kuesioner, tahap selanjutnya dilakukan analisis interval dengan memberikan skor pada setiap jawaban yang ada seperti ini[15]:

1. STS (Sangat Tidak Setuju) : 1 skor
2. TS (Tidak Setuju) : 2 skor
3. C (Cukup) : 3 skor
4. S (Setuju) : 4 skor
5. SS (Sangat Setuju) : 5 skor

Setelah memberikan bobot penilaian pada setiap jawaban peneliti melakukan perhitungan skor maksimum yang digunakan untuk menghitung nilai index dengan cara jumlah responden x skor tertinggi skor likert = 20 x 5 = 100. Selanjutnya dilakukan perhitungan data pada Tabel 3.

Tabel 4**.** Perhitungan skala likert

No	Jawaban					Total Skor	Nilai Indeks ((Total Skor / Skor Maksimum) x 100)%
	STS (1 Skor)	TS (2 Skor)	C (3 Skor)	S (4 Skor)	SS (5 Skor)		
1	-	-	-	40	50	90	90%
2	-	-	3	28	60	91	91%
3	-	-	3	36	50	89	89%
4	-	-	3	44	35	89	89%

Table 4**.** Perhitungan skala likert

Index Range	Hasil
0% - 19,99%	Sangat Tidak Setuju
20% - 39,99%	Tidak Setuju
40% - 59,99%	Kurang Setuju
60% - 79,99%	Setuju
80% - 100%	Sangat Setuju

Dapat dilihat pada Tabel 4 Bahwa setiap pertanyaan memiliki total nilai index sebesar 88% dan dapat digolongkan "Sangat Setuju" dengan melihat interval penilaian pada Tabel 5. Berdasarkan presentase nilai diatas dapat disimpulkan bahwa rancangan sistem sinkronisasi data menggunakan *Google Cloud Pub/Sub* dan *Flask* dinilai mampu menjaga data tersinkronisasi dan dapat memperbaiki data bermasalah yang tidak tersampaikan.

5. KESIMPULAN DAN SARAN

Berdasarkan hasil penelitian dapat disimpulkan bahwa sistem dinilai mampu melakukan sinkronisasi data *offline* ke *cloud* dan menjaga keabsahan data di *CloudSQL*. Selain itu fitur *Dead Letter Queue* pada *Google Cloud Pub/Sub* dapat membantu *Backend Developer* PT XYZ dalam melihat dan memperbaiki data yang tidak tersampaikan. Kesimpulan ini dapat didukung oleh hasil kuesioner yang melibatkan 20 *Backend Developer* di PT XYZ selaku pengguna sistem dan dari hasil tersebut rancangan sistem memperoleh persentase sebesar 88% dan digolongkan "Sangat Setuju" terhadap rancangan sistem yang dibuat.

Saran bagi penelitian selanjutnya adalah agar dapat memberikan *idempotent rest api* pada saat melakukan *publish* sehingga, *duplicate request* dapat memberikan response yang sama selain itu dapat dilakukan perbandingan tipe *push* dan *pull* yang ada pada *subscriber*.

DAFTAR PUSTAKA

[1] E. Riana, "Implementasi Cloud Computing Technology dan Dampaknya Terhadap Kelangsungan Bisnis Perusahaan Dengan Menggunakan Metode Agile dan Studi Literatur.," vol. *7*, no. 3, pp. 439-449, 2020.

[2] N. R. A. Salam and S. Ali, "Determining Factors of Cloud Computing Adoption: A Study of Indonesian Local Government Employees," *Journal of Accounting and Investment*, vol. 21, no. 2, pp. 312-333, 2020.

[3] P. K. Deny, Y. Mahmud, Widarti, D. W., *"*Implementasi Sinkronisasi Database Berbasis *RESTful Web Services* pada Aplikasi Presensi*,"* vol. 5, no. 1, pp. 01-08, 2020.

[4] P. Kumari and P. Kaur, "A survey of fault tolerance in cloud computing,"vol. 33, no. 10, pp. 1159-1176, 2021.

[5] R. Suci, F. Jannatin, A. Suharsono, and A. Bhawiyuga, "Implementasi Publish-Subscribe Pada Delay Tolerant Network (DTN)," vol. 1, no. 2, pp. 118-124, 2017.

[6] M. Kumar, "Google Cloud Platform: A Powerful Big Data Analytics Cloud Platform.," vol.4, no. 11, pp. 387-392, 2016.

[7] R. E. Putra, M. Izzati, and F. Dewi, "OPTIMASI KINERJA POINT OF SALE (POS) DENGAN PENERAPAN SINKRONISASI DATABASE

MENGGUNAKAN MIDDLEWARE," vol.12, no. 2, pp 123-129, 2017.

[8] R. Zumadilla Pratama, & H. Nurwarsito, "*Monitoring Penggunaan Daya Listrik menggunakan Protokol MQTT berbasis Web,"* Vol. 3, no.11, pp. 10820-10826, 2019.

[9] F. Nabilla, P. Irzan, D. P. Kartikasari, and A. Bhawiyuga, "Implementasi Sistem Tracking Posisi Ambulans pada Smart Dispatcher Menggunakan Metode Komunikasi Publish/Subscribe," vol. 4, no. 1, pp. 413-420, 2020.

[10] B. S. Alamsyah, B. Adhitya, & K. P. Dany, "*Implementasi Perangkat Mobile Publisher Subscriber Sebagai Perantara Pengiriman Data Sensor Dari Lapangan Ke Pusat Data.*," vol. 3, no. 2, pp. 1639-1647, 2019.

[11] A. T. S. Alim, D. P. Kartikasari, & F. A. Bakhtiar, "*Implementasi Paradigma Publish-Subscribe Untuk Menjalankan Event-Based Monitoring Pada Sistem Pengamatan Kandang Ternak,"* vol. 2, no. 10, pp. 3696-3702, 2020.

[12] Sugiyono, Metode Penelitian Kuantitatif Kualitatif dan R&D. Bandung: Alfabeta, 209AD.

[13] I. Barokah, & A. Asriyanik, "Analisis Perbandingan Serverless Computing Pada Google Cloud Platform.," vol. *7* no. 2, pp. 169–187, 2021.

[14] T. Hidayat & M. Muttaqin, "Pengujian Sistem Informasi Pendaftaran dan Pembayaran Wisuda Online menggunakan Black Box Testing dengan Metode Equivalence Partitioning dan Boundary Value Analysis.," vol. 6, no. 1, pp. 25-29, 2018.

[15] V. H. Pranatawijaya,Widiatry, W., , R. Priskila & P. B. A. A. Putra, "Penerapan Skala Likert dan Skala Dikotomi Pada Kuesioner Online.," vol *5* no. 2, pp. 128–137, 2019.