IMPLEMENTASI SISTEM *TRACKING* POSISI AMBULANS PADA *SMART DISPATCHER* MENGGUNAKAN METODE KOMUNIKASI *PUBLISH/SUBSCRIBE*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Farah Nabilla Putri Irzan
NIM: 165150207111052

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2020

PENGESAHAN

IMPLEMENTASI SISTEM *TRACKING* POSISI AMBULNAS PADA *SMART DISPATCHER* MENGGUNAKAN METODE KOMUNIKASI *PUBLISH/SUBSCRIBE*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Farah Nabilla Putri Irzan
NIM: 165150207111052

Skripsi ini telah diuji dan dinyatakan lulus pada
18 Februari 2020
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Dany Primanita Kartikasari, S.T., M.Kom.	Adhitya Bhawiyuga, S.Kom., M.Sc.
NIP: 19771116 200501 2 003	NIK: 19890720 201803 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D.
NIP: 19710518 200312 1 002

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar referensi.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 26 Februari 2020

Farah Nabilla Putri Irzan

NIM: 165150207111052

ABSTRAK

Farah Nabilla Putri Irzan, Implementasi Sistem Tracking Posisi Ambulans pada Smart Dispatcher Menggunakan Metode Komunikasi Publish/Subscribe

Pembimbing: Dany Primanita Kartikasari, S.T., M.Kom. dan Adhitya Bhawiyuga, S.Kom., M.Sc.

Ambulans merupakan kendaraan yang dilengkapi peralatan medis untuk mengangkut orang sakit atau korban kecelakaan. Pelacakan lokasi ambulans diperlukan untuk menentukan lokasi ambulans. Dengan ini dibangun sistem *tracking* posisi ambulans untuk melacak lokasi ambulans. Pada sistem ini digunakan aplikasi android untuk mengirimkan data ambulans, web server untuk menerima data ambulans dan aplikasi android pasien. Ambulans yang memiliki data lokasi mengirimkan data ke pusat penerima data untuk menyimpan lokasi ambulans dan melakukan perhitungan jarak terdekat terhadap ambulans-ambulans dan pasien. Pengiriman data antara entitas pengirim lokasi ambulans dan entitas penerima data menggunakan metode komunikasi *publish/subscribe*. Hasil pengujian rata-rata akurasi penentuan jarak dengan formula haversine yang digunakan pada sistem *tracking* ambulans adalah 99.88810212129279% dan rata-rata galat 0.1118978787072161%. Waktu respons pengiriman data *publisher* ke *subscriber* memiliki rata-rata waktu 880.2 *milliseconds.* Waktu respons mengirimkan permintaan dari aplikasi android pasien ke web server untuk penentuan ambulans dengan 1000 data ambulans pada basis data memiliki rata-rata waktu respons 70,06666667 *milliseconds*, 2000 data ambulans memiliki rata-rata waktu respons 109,6206897 *milliseconds*, 3000 data ambulans memiliki rata-rata waktu respons 143,7 *milliseconds*, 4000 data ambulans memiliki rata-rata waktu respons 187,9666667 *milliseconds*, dan 5000 data ambulans memiliki rata-rata waktu respons 238,2 *milliseconds*.

Kata kunci: tracking, ambulans, smart dispatcher, publish/subscribe

ABSTRACT

Farah Nabilla Putri Irzan, Implementation of Ambulance Position Tracking System on Smart Dispatcher Using Publish/Subscribe Communication Method.

Supervisors: Dany Primanita Kartikasari, S.T., M.Kom. and Adhitya Bhawiyuga, S.Kom., M.Sc.

Ambulance is a vehicle equipped with medical equipment to transport sick person or accident victims. Ambulance tracking is needed to determine the location of the ambulance. Therefore ambulance tracking system is built to track the location of the ambulance. In this system an android application is used to send ambulance data, a web server to receive ambulance data and an android application for patients. Ambulance entity sends data to the data receiver center to store the location of the ambulances and find nearest ambulance to the patient. Data transmission between data sender and the data receiver used publish/subscribe communication method. The results of accuracy testing of distance determination with haversine formula is 99.88810212129279% and the average error is 0.1118978787072161%. The response time for sending data from the publisher to the subscriber has an average time of 880.2 milliseconds. Response time sending requests from the patient's android application to the web server with 1000 ambulance data in the database has an average response time of 70.06666667 milliseconds, 2000 ambulance data has an average response time of 109.6206897 milliseconds, 3000 ambulance data has an average the average response time was 143.7 milliseconds, 4000 ambulance data had an average response time of 187.9666667 milliseconds, and 5000 ambulance data had an average response time of 238.2 milliseconds.

Keywords: tracking, ambulance, smart dispatcher, publish/subscribe

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Gawat Darurat merupakan keadaan klinis yang membutuhkan segera tindakan medis untuk menyelamatkan nyawa dan mencegah kecacatan. Menurut Peraturan Menteri Kesehatan Nomor 47 Tahun 2018 tentang Pelayanan Kegawatdaruratan, kegawatdaruratan memiliki kriteria mengancam nyawa, membahayakan diri dan orang lain/lingkungan, adanya gangguan pada jalan nafas, pernafasan, dan sirkulasi, adanya penurunan kesadaran, adanya gangguan hermodinamik dan/atau memerlukan tindakan segera. Pasien yang mengalami keadaan gawat darurat harus ditangani dengan cepat. Menurut Limantara, et al. (2015) pola kematian pada IGD Rumah Sakit Angkatan Laut Dr. Ramelan Surabaya menunjukan adanya pengaruh faktor usia, kegawatan, dan kecepatan penangan yang juga dipengaruhi oleh faktor prafasilitas. Penanganan prafasilitas adalah penanganan sebelum pasien sampai di fasilitas kesehatan. Evakuasi pasien dengan menggunakan ambulans dapat mengatasi masalah kecepatan evakuasi dan penanganan prafasilitas pasien.

Ambulans merupakan kendaraan yang dilengkapi peralatan medis untuk mengangkut orang sakit atau korban kecelakaan. Evakuasi medis menggunakan ambulans disertai dengan upaya menjaga stabilisasi dan resusitasi pasien oleh tenaga medis yang ikut serta dalam evakuasi pasien menggunakan ambulans. Resusitasi diperuntukan bagi pasien yang mengalami henti jantung ataupun pasien yang mengalami krisis tanda vital. Jarak ambulans dan pasien juga menjadi faktor kecepatan pasien sampai ke fasilitas kesehatan, karena jika jarak pasien dengan ambulans berjauhan, maka pasien akan semakin lama untuk ditangani oleh dokter di faslitas kesehatan. Saudin, Agnes dan Rini (2016) menyebutkan bahwa terdapat korelasi keterlambatan penanganan pasien stroke saat merujuk ke RSUD Jombang dengan jarak rujukan. Penentuan ambulans terdekat dari pasien dapat mengurangi waktu evakuasi sehingga keterlambatan penanganan pasien dapat dihindari. Untuk menentukan ambulans terdekat dari pasien, perlu diketahui terlebih dahulu lokasi dari ambulans-ambulans yang tersedia.

Pada penelitian sebelumnya dibuat *web-based GPS-GPRS vehicle tracking system*. Pada penelitian ini kendaraan mengirimkan lokasi koordinat yang diperoleh dari GPS melalui GPRS, dimana jaringan GSM akan meneruskan informasi ke server tujuan sebagai paket HTTP (Salim & Idrees, 2013). Naik (2017) menyebutkan bahwa protokol HTTP memiliki ukuran pesan paling besar dan membutuhkan daya dan sumberdaya paling tinggi diantara protokol AMQP, MQTT dan CoAP. Protokol HTTP didesain untuk web bukan untuk IoT, oleh karena itu HTTP membutuhkan ukuran pesan yang besar diantara protokol AMQP, MQTT dan CoAP.

Masalah yang ada adalah pasien membutuhkan ambulans dalam jangka waktu yang cepat. Ini dapat dicapai dengan mengimplementasikan sistem yang

dapat menentukan ambulans terdekat dan tersedia dari lokasi pasien. Sistem penentuan ambulans tersebut harus dapat melakukan reservasi ambulans sehingga pasien tidak mendapatkan ambulans yang sedang digunakan oleh orang lain.

Untuk dapat menyelesaikan permasalahan menentukan ambulans terdekat dari lokasi pasien dapat dilakukan dengan *tracking* posisi ambulans. *Tracking* posisi ambulans dilakukan untuk mengetahui posisi ambulans saat ini sehingga sistem dapat menentukan ambulans terdekat dengan pasien. Untuk menentukan lokasi ambulans, dapat menggunakan GPS. GPS (*Global Positioning System*) merupakan sistem navigasi yang menggunakan sinyal satelit. Lokasi dari ambulans dan pasien dapat ditentukan oleh koordinat, yaitu bilangan yang dipakai untuk menunjukan suatu titik. Koordinat didapatkan dari hasil perpotongan garis lintang (*latitude*) dengan garis bujur (*longitude*).

Metode komunikasi *publish/subscribe* cocok untuk mendistribusikan data secara *realtime* karena *publisher* mengirimkan data segera ke *subscriber* dan *publisher* dapat langsung mengirimkan data tanpa perlu menunggu balasan (*acknowledgment*) dari *subscriber*. *Publish/subscribe* cocok digunakan bila pembaruan data langsung dikirimkan ke *subscriber* (Oh, et al., 2009). Pendistribuatian data secara *realtime* dibutuhkan pada sistem *tracking* posisi ambulans untuk mengetahui posisi ambulans saat ini.

Salah satu protokol yang menerapkan metode komunikasi *publish/subscribe* adalah protokol MQTT. MQTT adalah singkatan dari *Message Queuing Telemetry Transport*. Dizdarevic, et al. (2018) mengungkapkan karena kesederhanaan protokol MQTT dan *header* pesan yang kecil yaitu 2 byte *header* tetap, dibandingankan dengan *messaging protocol* lainnya, MQTT sering direkomendasikan sebagai solusi komunikasi pilihan pada IoT. Karena *header* yang ringan, MQTT membutuhkan daya yang lebih rendah . Atas dasar tersebut, penelitian ini mengggunakan metode komuniskasi *publish/subscribe* dengan protokol MQTT untuk mengirimkan data ambulans.

Pada peneliian ini terdapat 3 komponen yaitu pengirim data ambulans, pusat penerima data dan aplikasi untuk pasien. Protokol MQTT diimplementasikan pada pengirim data ambulans dan pusat penerima data, dimana pengirim data ambulans adalah *publisher* dan pusat penerima data adalah *subscriber*. *Publisher* diimplementasikan pada aplikasi android. Data yang dikirimkan oleh *publisher* adalah lokasi ambulans dan status ambulans. Lokasi ambulans didapatkan dari GPS dan status ambulans didapatkan dari masukan pengguna. *Subscriber* diimplementasikan pada web server. Web server dapat menerima *request* dari aplikasi pasien untuk mendapatkan ambulans yang dibutuhkan oleh pasien. Pada web server, data terbaru lokasi dan status ambulans disimpan pada basis data.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka dapat diuraikan rumusan masalah yang mendasari penelitian ini, diantaranya adalah sebagai berikut:

1. Bagaimana hasil implementasi metode komunikasi publish/subscribe untuk mengetahui posisi ambulans dan statusnya?
2. Bagaimana algoritma pemilihan ambulans terdekat dari posisi pasien?
3. Berapa lama waktu respons *publisher* mengirimkan data lokasi ke *subscriber*?
4. Bagaimana akurasi lokasi yang didapatkan dari sistem *tracking*?

1.3 Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan metode komunikasi publish/subscribe untuk mengetahui posisi ambulans dan statusnya.
2. Mengetahui algoritma pemilihan ambulans terdekat dari posisi pasien.
3. Mengetahui waktu respons *publisher* mengirimkan data lokasi ke *subscriber*.
4. Mengetahui akurasi lokasi yang didapatkan dari sistem *tracking*.

1.4 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Sebagai solusi untuk mendapatkan lokasi ambulans terdekat dari jarak pasien.
2. Dapat merancang dan menerapkan metode komunikasi *publish/subscribe* dengan menambahkan sistem pada sisi subscriber yang melakukan layanan (*service*) untuk melakukan komputasi.

1.5 Batasan Masalah

Batasan Masalah dari penelitian ini adalah sebagai berikut:

1. Node ambulans (*publisher*) yang digunakan merupakan aplikasi perangkat bergerak.
2. Aplikasi perangkat bergerak pasien mengirimkan lokasi perangkat tersebut.
3. Jarak hanya diukur berdasarkan perhitungan posisi *latitude* dan *longitude* yang didapatkan dari sensor GPS.

1.6 Sistematika Pembahasan

Sistematika pembahasan menjelasan mengenai hal-hal yang akan dibahas dalam penelitian ini dan menjelaskan mengenai penyusunan penulisan laporan

serta deskripsi singkat dari masing-masing penyusun. Penelitian ini disusun sebagai berikut:

BAB I PENDAHULUAN

Bab pendahuluan adalah bagian yang menjelaskan hal yang melatarbelakangi penelitian ini. Dari masalah tersebut kemudian dibuat rumusan masalah dan selanjutnya mendapatkan tujuan dan manfaat dalam penelitian. Dalam melakukan penelitian juga terdapat batasan –batasan.

BAB II LANDASAN KEPUSTAKAAN

Pada bab landasan kepustakaan berisi pembahasan tentang teori-teori, metode, model atau informasi yang berkaitan dengan penelitian yang sebagai landasan dalam penelitian.

BAB III METODOLOGI

Pada bab metodologi menjelaskan mengenai metode, teknik, atau langkah-langkah yang digunakan dalam penelitian ini.

BAB IV ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM

Pada bab analisis kebutuhan perancangan sistem dijelaskan mengenai perancangan sistem yang akan dibuat pada penelitian ini dan kebutuhan apa saja yang dibutuhkan dalam sistem.

BAB V IMPLEMENTASI SISTEM

Pada bab ini akan menjelaskan mengenai implementasi dari sistem yang telah dirancang.

BAB VI PENGUJIAN SISTEM

Pada bab pengujian sistem akan melakukan pengujian terhadap penelitian yang dilakukan setelah implementasi berhasil dilakukan.

BAB VII PENUTUP

Bab penutup berisi kesimpulan dan saran dari penulis. Kesimpulan didapat dari hasil pengujian dan analisis penelitian.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Kajian Pustaka

Penelitian dengan judul "*Smart Ambulance System*" mengusulkan ide untuk menyelamatkan pasien secepat mungkin dengan *smart ambulance system*. Ide tersebut dapat menguntungkan pengguna ketika terjadi keadaan gawat darurat. Pada penelitian ini mengusulkan arsitektur sistem dimana terdapat aplikasi android yang dapat mengirimkan permintaan untuk mengetahui informasi ambulans dan rumah sakit ke server. Pada arsitektur sistem penelitian ini, alat-alat pada ambulans akan mengirimkan informasi kesehatan pasien ke rumah sakit yang dituju (Gupta, et al., 2016). Dari penelitian ini dapat disimpulkan bahwa sistem "Smart Ambulance System" bermanfaat ketika terjadi kasus gawat darurat. Sistem ini dapat membantu menyediakan layanan medis yang lebih cepat.

Penelitian "*Design and Evaluation of an IoT enabled Secure Multi-service Ambulance Tracking System*" membuat sistem *tracking* ambulans untuk menyediakan informasi ambulans secara *realtime* kepada pasien dan rumah sakit. Pada penelitian ini mengimlementasikan AES-CCM untuk menyediakan kerahasiaan, otentikasi dan integritas data. Sistem ini dibagi menjadi 3 bagian yaitu *cloud server*, modul IoT dan aplikasi android. Modul IoT mendapatkan lokasi dengan GPS kemudian mengirimkannya ke *cloud server* melalui internet dengan GPRS. (Sarbpreet, et al., 2016).

Penelitian dengan judul "*Design and Implementation of Web-Based GPS-GPRS Vehicle Tracking System*" mengusulkan sistem tracking kendaraan berbasis GPS-GPRS dimana GPS untuk menentukan lokasi kendaraan dan GPRS untuk mengirimkan data ke server. Jaringan GSM akan meneruskan informasi ke server tujuan sebagai paket HTTP (Salim & Idrees, 2013). Pada penelitian ini, integrasi GPS dan GPRS dapat menyediakan pelacakan secara realtime dengan mengirimkan data dengan menggunakan protokol HTTP.

Pada penelitian dengan judul "*A Low Cost Implementation of MQTT Using ESP32*" mengimplementasikan gagasan untuk *smart home*, sistem lampu jalan untuk *smart cities*, sistem peringatan kebakaran, pemantauan cuaca dan sebagainya. Pada penelitian ini terlihat bahwa komunikasi antara *low power* ESP 8266 WiFi sebagai klien dengan klien pada *smartphone* dan laptop menggunakan protokol MQTT menjadi lebih mudah dan lebih dapat diandalkan. Klien pada *smartphone* dan laptop yang melakukan *subscribe* ke topik *temperature, humidity* dan *light intensity level* mendapatkan pembaruan. (Kodali & Mahesh, 2016)

2.2 Dasar Teori

2.2.1 *Tracking*

Tracking dalam Bahasa Indonesia berarti pelacakan. *Tracking* dilakukan untuk mengamati objek yang bergerak. *Tracking* saat ini banyak digunakan, seperti untuk melacak anak, melacak aset, melacak kendaraan-kendaraan, atau melacak peralatan.

Mukhtar (2015) menyebutkan bahwa *tracking* dibagi menjadi 4 jenis, yaitu pelacakan untuk menemukan arah, teknik pelacakan loran, pelacakan plang (*signpost*) dan pelacakan berbasis GPS. Pelacakan dengan teknik loran digunakan untuk menemukan lokasi dari sebuah objek dengan menggunakan perbedaan waktu (*time difference*) antara sinyal radio dari dua atau lebih transmitter. Pelacakan plang (*signpost*) biasanya dilakukan pada jalur kereta api.

2.2.2 *Global Positioning System*

GPS adalah singkatan dari *Global Positioning System*. GPS merupakan sistem navigasi menggunakan sinyal satelit. GPS dapat menentukan letak dipermukaan bumi.

Sistem GPS dasar terdiri dari dua segmen yang bertujuan untuk menyediakan pelacakan yang konsisten dan andal dan layanan waktu. GPS menggunakan paling tidak 3 satelit untuk melacak objek yang ada di bumi secara akurat. Satelit tersebut memancarkan 2 sinyal, yaitu pada frekuensi 1575.42 MHz dan frekuensi 1227.6 MHz (Mulla, et al., 2015).

2.2.3 *Smart Dispatcher*

Ambulance dispatching (Pengiriman ambulans) pada *Emergency Medical Service* (EMS) menentukan ambulans terhadap panggilan ambulans sehingga waktu respons dapat diminimalkan. *Greedy policy* merupakan kebijakan yang paling sering digunakan pada EMS. *Greedy policy* mengirim ambulans terdekat yang tersedia. (Lee, 2014)

Samani dan Zhu (2016) menyebutkan bahwa konsep mesin berteknologi tinggi yang dapat melayani dengan baik atau dapat membebaskan manusia dari tugas telah menjadi objek imajinasi manusia, dapat dilihat dari banyaknya pekerjaan yang telah tergantikan oleh otomatisasi. *Smart world* diharapkan dapat melibatkan pengindraan, komputasi dan komunikasi dimana-mana.

Smart dispatcher berarti operator cerdas. *Smart dispatcher* melibatkan otomatisasi dalam merespon panggilan darurat atau tidak darurat untuk mendapatkan bantuan dan informasi. *Smart dispatcher* melakukan pengindraan seperti menentukan lokasi dan melakukan komputasi untuk menentukan respon terhadap panggilan.

2.2.4 *Publish/Subscribe*

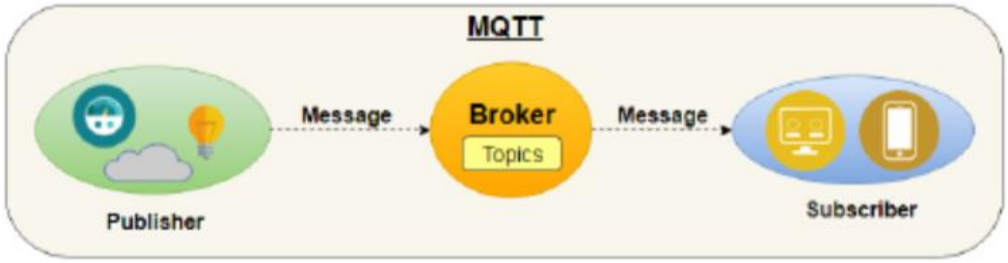
Publish/subscribe merupakan metode komunikasi. Pada *publish/subscribe* terdapat *publisher*, *subscriber* dan *broker*. *Publisher* adalah entitas yang menyediakan data. *Subscriber* adalah entitas yang menerima data. *Broker* adalah perantara antara *publisher* dan *subscriber*, *broker* memilih *subscriber* yang tepat untuk mendapatkan data dari *publisher*. Dari penjelasan mengenai *publisher*, *subscriber* dan *broker*, dapat dikatahui bahwa *publisher* tidak mengirimkan data secara langsung ke *subscriber*, melainkan melalui *broker*, dan *broker* yang akan memilih kepada *subscriber* mana data akan dikirimkan. Sehingga *publisher* dapat fokus untuk menghasilkan informasi dan *subscriber* untuk menggunakan data tersebut, bukan pada melacak satu sama lain antara *publisher* dan *subscriber* (Banerjee & Sahni, 2015).

Dizdarevic, et al. (2018) menyebutkan bahwa metode komunikasi *publish/subscribe* memiliki beberapa kelebihan yaitu *publisher* dan *subscriber* tidak perlu untuk mengetahui keberadaan satu sama lain. Keuntungan lainnya adalah satu *subscriber* dapat menerima informasi dari *publisher* yang berbeda dan satu *publisher* dapat mengirimkan data ke banyak *subscriber* (mendukung komunikasi *many-to-many*). Serta *publisher* dan *subscriber* tidak perlu aktif dalam waktu yang sama untuk bertukar informasi, karena broker dapat menyimpan pesan untuk klien yang sedang tidak terhubung.

Beberapa contoh protokol yang mengimplemenasikan metode komunikasi publish subscribe adalah MQTT, AMQP dan DDS. MQTT adalah singkatan dari *Message Queuing Telemetry Transport*, AMQP adalah singkatan dari *Advanced Message Queuing Protocol*, dan DDS adalah singkatan dari *Data Distributed Service*.

2.2.5 *Message Queuing Telemetry Transport* (MQTT)

MQTT merupakan salah satu protokol yang dapat digunakan untuk komunikasi pada sistem komputer. MQTT berjalan diatas TCP/IP. Protokol MQTT didesain dengan karakteristik sederhana, ringan dan mudah untuk diimplementasikan (Tarigan, Sitepu, & Hutagalung, 2014). Protokol MQTT menggunakan sistem *publish/subscribe,* dimana *publisher* merupakan pengirim data dan *subscriber* merupakan penerima data. Pada protokol MQTT, klien merupakan publisher atau subscriber terhubung dengan server atau broker. Gambar 2.1 berikut menampilkan arsitektur MQTT.



Gambar 2. 1 Arsitektur MQTT

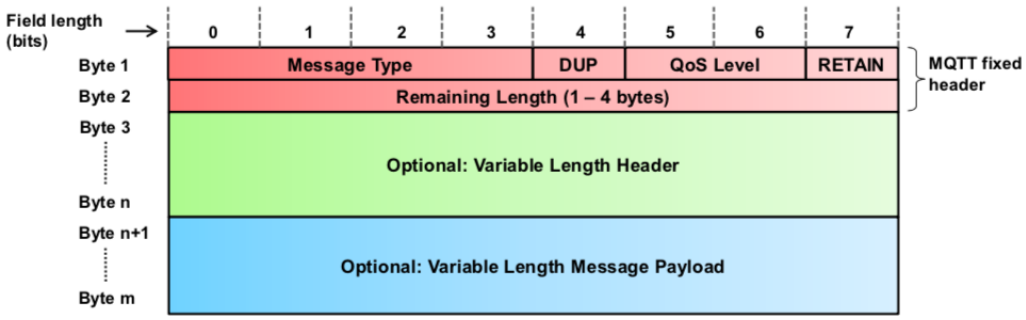
Sumber : Ansari, Rehman, dan Mughal (2018)

Broker pada MQTT berfungsi untuk menangani data dan distribusi data tersebut. Distribusi pada broker dilakukan berdasarkan dengan topik. *Publisher* mengirimkan data dengan topik tertentu, dan *subscriber* berlangganan pada topik tertentu. Broker akan mendistribusikan data yang diterima dari *publisher* ke *subscriber* berdasarkan topik yang diinginkan oleh *subscriber*.

Komponen infromasi pada MQTT adalah topik, *payload/message*, dan *Quality of Service*. Topik merupakan label untuk informasi yang dikirimkan atau diterima. *Payload* merupakan informasi yang dikirimkan atau diterima dan QoS merupakan jaminan ketersampaian informasi.

Pada MQTT terdapat 3 level QoS. QoS adalah *quality of service*, yang merupakan kemampuan jaringan untuk menyediakan layanan. 3 QoS tersebut adalah, yaitu QoS level 0 , Qos level 1 dan QoS level 2. Berikut merupakan penjelasan mengenai masing-masing level QoS:

- Qos level 0 adalah *at most once delivery*, yang berarti pesan akan dikirimkan hanya sekali, tidak ada jaminan pesan tersebut sampai ke broker.
- QoS level 1 adalah at *least once deliv*ery, yang berarti paling tidak pesan dikirimkan sekali. Jika pesan tidak diterima broker, maka publisher akan mengirim kembali. Pada QoS level 1 terdapat kemungkinan terjadi duplikasi pesan.
- QoS level 2 adalah *exactly once delivery*, yang berarti pesan akan dikirimkan tepat sekali, tidak ada pesan duplikat yang diterima.



Gambar 2. 2 Format Pesan MQTT

Gambar 2.2 menunjukan format pesan MQTT. *Message type* adalah tipe pesan pada MQTT yang disebutkan pada Tabel 2.1. DUP adalah *duplicate message flag*, DUP untuk menentukan apakah pesan tersebut merupakan pesan duplikat dan dikirim ulang karena penerima yang dituju tidak menjawab pesan asli. QoS level adalah level *quality of service*. *Retain* untuk menentukan apakah pesan tersebut menginstruksikan server untuk mempertahankan pesan publish yang terakhir diterima dan mengirimkannya sebagai pesan pertama ke *subscriber* baru.

Tabel 2. 1 Tipe Pesan MQTT

Nilai	Pesan MQTT
1	CONNECT
2	CONACK
3	PUBLISH
4	PUBACK
5	PUBREC
6	PUBREL
7	PUBCOMP
8	SUBSCRIBE
9	SUBACK
10	UNSUBSCRIBE
11	UNSUBACK
12	PINGREQ
13	PINGRESP
14	DISCONNECT

2.2.6 Formula Haversine

Formula haversine adalah formula yang digunakan untuk menghitung jarak antara dua titik berdasarkan posisi garis lintang (*latitude*) dan garis bujur (*longitude*). Persamaan 2.1 berikut merupakan formula haversine.

$$d = 2r \sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (2.1)$$

dimana:

d : jarak kedua titik dalam satuan kilometer

r : radius bumi (6371 km)

φ_1 : garis lintang titik pertama dalam radian

φ_2 : garis lintang titik kedua dalam radian

$\Delta\varphi$: garis lintang 1 – garis lintang 2

$\Delta\lambda$: garis bujur 1 – garis bujur 2

BAB 3 METODOLOGI PENELITIAN

Pada bab metodologi peneitian ini menjelaskan tahapan-tahapan yang dilakukan untuk mengimplementasikan sistem *tracking* posisi ambulans pada *smart dispatcher* menggunakan metode komunikasi *pubish*/*subscribe*. Tahapan-tahapan tersebut ditunjukan seperti pada Gambar 3.1.

Gambar 3. 1 Diagram Alir Tahapan Penelitian

Penelitian ini menggunakan tipe penelitian implementatif perancangan. Penelitian implementatif adalah tipe penelitian yang menghasilkan produk sebagai solusi terhadap permasalahan dalam penelitian. Pendekatan pada tipe penelitian implementatif yang digunakan pada penelitian ini adalah pendekatan perancangan (*design*). Tipe penelitian implementasi perancangan adalah penelitian yang melakukan proses dari analisis perancangan hingga pengujiannya. Tahapan dari penelitian ini dimulai dari studi literatur hingga kesimpulan dan saran.

Tahapan pertama adalah studi literatur. Studi literatur adalah tahapan untuk mendapatkan referensi teori yang berhubungan dengan permasalahan yang akan

diteliti. Studi literatur diperlukan untuk mendalami hal-hal yang berhubungan dengan permasalahan. Setelah tahap studi literatur, dilakukan analisis kebutuhan. Analisis kebutuhan membahas mengenai kebutuhan sistem. Analisis kebutuhan sistem dibutuhkan dalam pengembangan sistem agar pengembangan sistem dapat lebih terarah dan sistematis. Tahapan selanjutnya adalah perancangan sistem. Perancangan sistem dilakukan dengan merancang sistem yang telah dideskripsikan kebutuhannya pada analisis kebutuhan. Sistem yang telah dirancang selanjutnya diimplementasikan pada tahap implementasi. Sistem yang telah diimplementasikan akan diuji untuk memastikan implementasi yang dibuat telah sesuai dengan kebutuhan. Dan terakhir tahap pengambilan kesimpulan dan saran. Kesimpulan akan menjawab rumusan masalah penelitian ini dan memberikan saran untuk pengembangan penelitian selanjutnya.

3.1 Studi Literatur

Studi literatur dilakukan dengan meninjau penelitian sebelumnya dan mempelajari teori-teori yang berkaitan dengan penelitian yang akan dilakukan yaitu implementasi sistem *tracking* posisi ambulans pada *smart dispatcher* menggunakan metode komunikasi *publish*/*subscribe*. Sumber teori penelitian ini adalah jurnal dan *proceedings*. Dasar teori tersebut adalah sebagai berikut:

1. *Publish*/*Subscribe*

   Dasar teori mengenai metode komunkasi *publish*/*subscribe* dibutuhkan untuk mengetahui bagaimana *publish*/*subscribe* bekerja, keuntungan menggunakan metode komunikasi *publish*/*subscribe* dan protokol yang menggunakan metode komunikasi *publish*/*subscribe*. Cara kerja metode komunikasi *publish*/*subscribe* dibutuhkan untuk memahami cara kerja metode komunikasi *publish*/*subscribe* sehingga dapat mengimplementasikan metode komunikasi tersebut.

2. Protokol MQTT (*Message Queuing Telemetry Transport*)

   Dasar teori mengenai protokol MQTT dibutuhkan untuk melakukan implementasi protokol tersebut. Dasar teori protokol MQTT untuk mengetahui dan memahami komponen apa saja yang untuk mengimplementasikan protokol MQTT. Komponen protokol MQTT seperti *publisher*, *subscriber* dan *broker*.

3. Formula Haversine

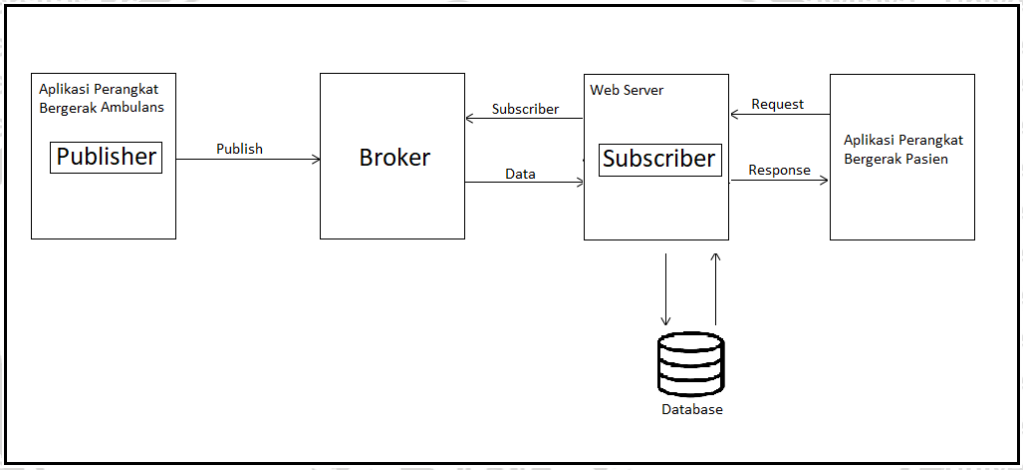
   Dasar teori mengenai formula haversine dibutuhkan untuk melakukan perhitungan jarak antara dua titik. Formula haversine memungkinkan untuk menghitung jarak antara dua titik koordinat dan menghasilkan jarak dalam satuan kilometer. Perhitungan jarak dibutuhkan untuk mengetahui jarak ambulans dan pasien sehingga dapat ditemukan ambulans terdekat dari pasien.

3.2 Analisis Kebutuhan

Analisis kebutuhan sistem dibutuhkan dalam pengembangan sistem agar pengembangan sistem dapat lebih terarah dan sistematis. Pada analisis kebutuhan sistem akan membahas mengenai kebutuhan fungsional. Kebutuhan fungsional adalah kebutuhan yang berisi layanan apa saja yang harus disediakan oleh sistem.

3.3 Perancangan Sistem

Pada perancangan sistem akan menjelaskan rancangan sistem yang akan dibuat dan perancangan pengujian. Gambar 3.2 menggambarkan sistem yang akan dibuat pada penelitian ini. Dari gambar tersebut dapat dilihat bahwa terdapat komponen *publisher*, *subscriber* dan aplikasi perangkat bergerak pasien. Komponen-komponen tersebut akan dirancang pada tahap perancangan sistem.



Gambar 3. 2 Diagram Blok Sistem

Perancangan dalam penelitian ini dibagi menjadi tiga bagian yaitu:

1. Perancangan Node *Publisher*

Node publisher merupakan node yang megirimkan data ambulans. Perancangan node *publisher* menjelaskan apa saja yang akan dilakukan node *publisher*. Pada perancangain node *publisher* akan dijelaskan rancangan pengiriman data dan rancangan data yang akan dikirimkan.

2. Perancangan Node *Subscriber*

Node *subscriber* merupakan node yang menerima data ambulans. Node *subscriber* merupakan *web server* yang menerima *request* dari aplikasi pasein. Perancangan node *subscriber* menjelaskan apa saja yang akan dilakukan pada node *subscriber*. Pada perancangan node *subscriber* akan dijelaskan rancangan menerima data, dan rancangan cara penentuan ambulans.

3. Perancangan Node Aplikasi Perangkat Bergerak Pasien

Node aplikasi perangkat bergerak pasein merupakan node yang digunakan pasein untuk mendapatkan ambulans. Node aplikasi perangkat bergerak pasien

mengirimkan *request* ke web server untuk mendapatkan ambulans. Perancangan node aplikasi perangkat bergerak pasein menjelaskan apa saja yang akan dilakukan pada node aplikasi perangkat bergerak pasein. Pada perancangan node aplikasi perangkat bergerak pasein akan dijelaskan rancangan melakukan *request* ambulans ke web server.

4. Perancangan Basis Data

Perancangan basis data membahas pengaturan basis data yang akan diimplementasikan pada sistem. Basis data diimplementasikan pada web server.

5. Perancangan Pengujian

Perancangan pengujian membahas pengujian yang dilakukan untuk menguji sistem. Pada perancangan pengujian akan dijabarkan scenario pengujian fungsional untuk menguji kebutuhan fungsional dan prosedur untuk melakukan pengujian akurasi dan response time. Scenario pengujian dirancang untuk menguji satu persatu kebutuhan fungsional.

3.4 Implementasi

Implementasi akan menjelaskan mengenai proses implementasi sistem *tracking* posisi ambulans pada *smart dispatcher* menggunakan metode komunikasi *publish*/*subscribe*. Implementasi pada pada penelitian ini dibagi kedalam tiga bagian yaitu implementai node *publisher*, implementasi node *subscriber* dan implementasi aplikasi perangkat bergerak pasien.

1. Implementasi Node *Publisher*

Node *pubisher* diimplementasikan pada aplikasi android dimana aplikasi tersebut dapat menerima masukan dari pengguna untuk memulai dan menghentikan pengirimkan data serta user dapat memasukan status dari ambulans. Implementasi *publisher* pada penelitian ini mencakup implementasi mendapatkan lokasi, implementasi memulai dan menghentikan *publish*, implementasi menghubungkan *publisher* dengan *broker*, dan implementasi *publish*.

2. Implementasi Node *Subscriber*

Node *subscriber* merupakan *web server* yang mengimplementasikan protokol MQTT, dapat menyimpan data pada basis data dan menerima *request* dari aplikasi perangkat bergerak pasien. Implementasi *subscriber* pada penelitian ini mencakup implementasi MQTT pada *web server*, implementasi menerima *request* dari aplikasi pasien, dan implementasi layanan penentuan ambulans.

3. Implementasi Node Aplikasi Perangkat Bergerak Pasien

Node aplikasi perangkat bergerak pasien diimplementasikan pada aplikasi android dimana aplikasi tersebut megirimkan permintaan untuk mendapatkan

ambulans. Implementasi aplikasi perangkat bergerak pasien pada penelitian ini mencakup implementasi mengirimkan *request* ke web server.

3.5 Pengujian Sistem

Pengujian dilakukan setelah melakukan implementasi, hal ini bertujuan untuk menguji sistem yang dibuat apakah memenuhi kebetuhan fungsional dan untuk mengetahui kinerja dari sistem. Pengujian sistem meliputi pengujian fungsional dan pengujian kinerja. Pengujian fungsional dilakukan didasari oleh kebutuhan fungsional. Pengujian kinerja dilakukan dengan *response time* yang dibutuhkan untuk menentukan ambulans yang sesuai dengan lokasi dan kasus pasien.

3.6 Kesimpulan dan Saran

Tahapan pengambilan kesimpulan dan saran memuat kesimpulan dan saran terhadap penelitian yang dilakukan. Kesimpulan didapatkan berdasarkan hasil perancangan, implementasi dan pengujian. Saran bertujuan untuk memberikan saran mengenai pengembangan yang bisa dilakukan dari penelitian selanjutnya.

BAB 4 ANALISIS KEBUTUHAN DAN PERANCANGAN SISTEM

Bab ini menjelaskan mengenasi analisis dan perancangan sistem. Analisis kebutuhan sistem membahas mengenai kebutuhan sistem. Perancangan sistem merupakan merancang sistem yang telah dideskripsikan kebutuhannya pada analisis kebutuhan sistem.

4.1 Analisis Kebutuhan Sistem

Analisis kebutuhan sistem dibutuhkan dalam pengembangan sistem agar pengembangan sistem dapat lebih terarah dan sistematis. Pada analisis kebutuhan sistem akan membahas mengenai kebutuhan sistem. Tabel 4.1 adalah kebutuhan fungsional pada sistem.

Tabel 4. 1 Kebutuhan Fungsional

No	Kode Fungsi	Kebutuhan Fungsional
1	KF-01-01	Node *publisher* harus dapat menentukan lokasi dari perangkat node menggunakan GPS
2	KF-01-02	Node *publisher* harus dapat melakukan *publish* data secara terus menerus ke broker dengan format JSON
3	KF-01-03	Node publisher harus dapat menerima informasi pasien yang membutuhkan ambulans
4	KF-01-04	Node publisher harus dapat menolak atau manerima melakukan penjemputan pasien
5	KF-02-01	Node *subscriber* harus dapat melakukan *subscribe* ke broker
6	KF-02-02	Node *subscriber* harus dapat menerima data dari *publisher* yang diteruskan oleh broker
7	KF-02-03	Node *subscriber* harus dapat melakukan pembaruan data yang diterima dari *publisher* pada basis data
8	KF-02-04	Node *subscriber* harus dapat melakukan penentuan ambulans yang dibutuhkan oleh pasien berdasarkan status ambulans, kasus pasien, dan lokasi ambulans
9	KF-02-05	Node *subscriber* dapat menerima *request* dari node aplikasi perangkat bergerak pasien untuk memberikan *response* berupa

		ambulans yang sesuai dengan lokasi pasien dan kasus pasien
10	KF-02-06	Node *subscriber* dapat mengirimkan informasi pasien ke node *publisher*
11	KF-03-01	Node aplikasi perangkat bergerak pasien dapat mengirimkan *request* ke node *subscriber* untuk mendapatkan ambulans yang sesuai dengan lokasi pasien dan kasus pasien

4.2 Perancangan Sistem

Perancangan sistem merupakan tahapan dimana komponen-komponen yang dibutuhkan oleh sistem dirancang. Perancangan sistem dilakukan agar saat mengimplementasikan sistem menjadi terarah karena sudah melakukan perancangan sistem yang akan diimplementasikan. Perancangan sistem meliputi gambaran umum sistem, perancangan alur sistem, perancangan node *publisher*, perancangan node *subscriber*, perancangan node aplikasi perangkat bergerak pasien, perancangan basis data dan perancangan pengujian.

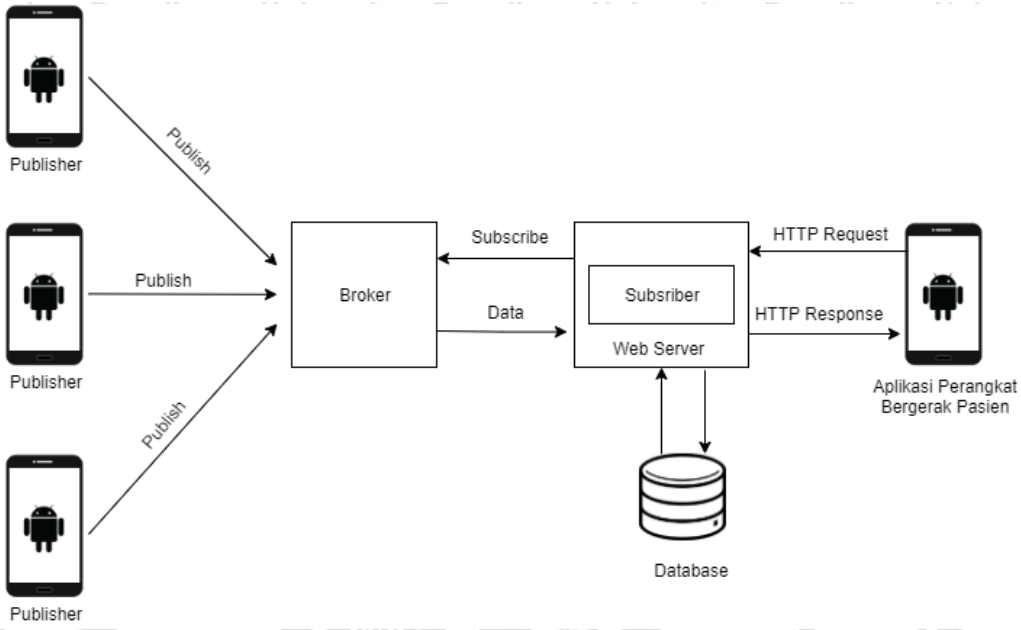
4.2.1 Gambaran Umum Sistem

Sistem yang akan dibuat pada penelitian ini adalah sistem *tracking* ambulans pada *smart dispatcher* menggunakan metode komunikasi *publish/subscribe*. Pada sistem terdapat node *publisher,broker,* node *subscriber* dan node aplikasi perangkat bergerak pasien.

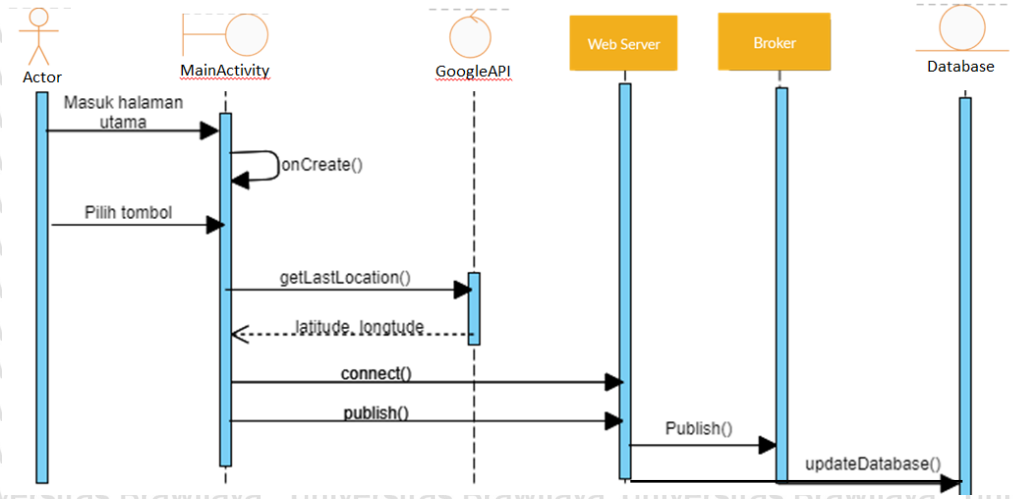
Node *publisher* merupakan aplikasi perangkat bergerak android untuk pengguna yang menjalankan ambulans. Node *publisher* mengirimkan data ambulans secara terus menerus ke broker. Data yang dikirimkan oleh node *publisher* adalah lokasi ambulans dan status ambulans. Node *publisher* mengirimkan data ambulans dengan protokol MQTT. Node *publisher* yang melakukan *publish* akan diteruskan oleh *broker* ke *subscriber*. Node *publisher* sebagai aplikasi perangkat bergerak untuk ambulans dapat menerima informasi permintaan penjemputan pasien dan melakukan konfirmasi menerima atau menolak permintaan penjemputan pasien.

Node *subscriber* merupakan web server yang dapat melakukan *subscribe* topik dan menerima data yang di *publish* oleh *publisher*. Node *subscriber* sebagai web server memiliki fungsi untuk menerima permintaan pengiriman ambulans oleh aplikasi perangkat bergerak pasien. Setelah menerima permintaan pengiriman ambulans, web server akan mengirimkan informasi pasien ke node publisher yang merupakan aplikasi perangkat bergerak. Node publisher yang menerima informasi pasien, akan mengirimkan konfirmasi untuk menolak atau menerima melakukan penjemputan pasien.

Node aplikasi perangkat bergerak pasien merupakan aplikasi perangkat bergerak android yang digunakan oleh pengguna pasien untuk mendapatkan ambulans. Node aplikasi perangkat bergerak pasien dapat melakukan permintaan pengirimaan ambulans dengan mengirimkan informasi pasien ke web server. Perancangan sistem yang dibangun akan ditunjukan pada Gambar 4.1.



Gambar 4. 1 Rancangan Arsitektur Sistem



Gambar 4. 2 Sequence Diagram Sistem

4.2.2 Perancangan Node *Publisher*

Node *publisher* merupakan aplikasi perangkat bergerak android untuk pengguna yang menjalankan ambulans. Node *publisher* mengirimkan data ambulans secara terus menerus ke broker. Data yang dikirimkan merupakan id ambulans, titik koordinat garis bujur lokasi ambulans, titik koordinat garis lintang ambulans dan status ambulans. Id ambulans merupakan nomor unik yang

membedakan antar ambulans satu dan ambulans lainnya untuk mengidentifikasi ambulans yang mengirimkan data. Titik koordinat garis lintang dan titik koordinat garis bujur untuk menentukan lokasi ambulans. Titik koordinat didapatkan dengan menggunakan GPS. Status ambulans merupakan keadaan ambulans ketika mengirimkan data, status ambulan dikirimkan untuk mengetahui apakah ambulans siap digunakan yang akan ditunjukan pada Tabel 4.2. Node *publisher* mengirimkan data ambulans dengan protokol MQTT. Node *publisher* yang melakukan *publish* akan diteruskan oleh *broker* ke *subscriber*.

Node *publisher* sebagai aplikasi perangkat bergerak untuk ambulans dapat menerima informasi permintaan penjemputan pasien dari node subscriber yang merupakan web server. Node *publisher* melakukan konfirmasi menerima atau menolak permintaan penjemputan pasien.

Tabel 4. 2 Status Ambulans

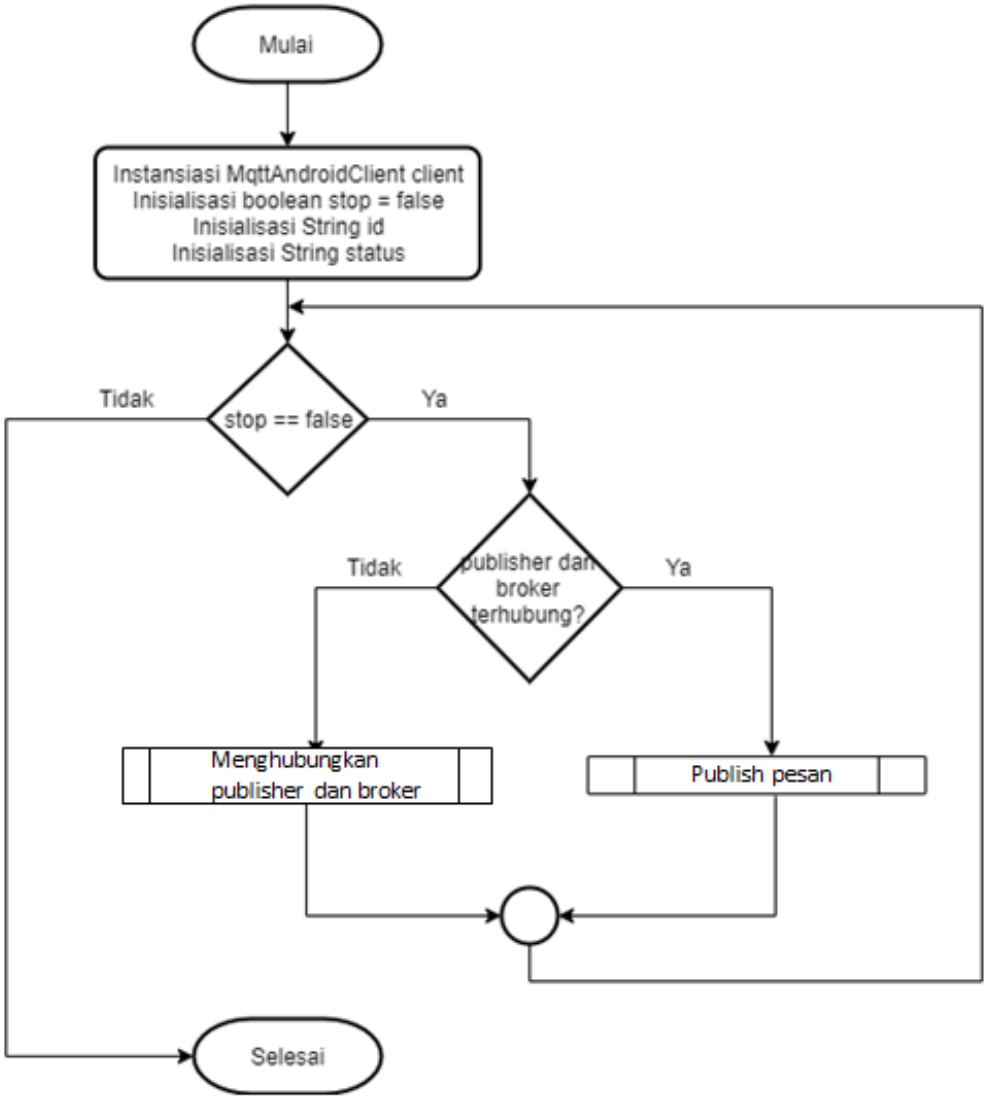
Status	Keterangan
ON DUTY	Status on duty berarti ambulans sedang bertugas sehingga tidak bisa digunakan untuk menjemput pasien lain.
OFF DUTY	Status off duty berati ambulans sedang tidak bertugas sehingga tidak bisa digunakan untuk menjemput pasien.
STAND BY	Status stand by berarti siap sedia. Ambulans tersedia untuk menjemput pasien.

Data ambulans tersebut akan dikirimkan dalam format JSON (Javascript Object Notation). Berikut merupakan perancangan data JSON yang dikirimkan dari *publisher* yang akan diteruskan oleh *broker* ke *subscriber*. Data JSON yang akan dikirimkan merupakan JSON object dengan *key* dan *value* yang ditunjukan pada Tabel 4.3.

Tabel 4. 3 Data JSON Object

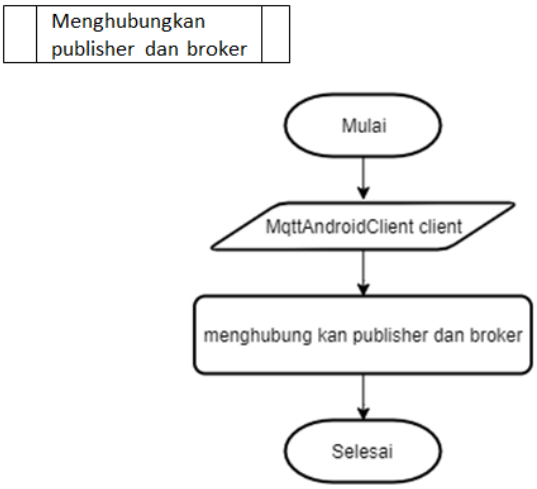
Tipe Data	Key	Value
number	id	Nilai dari id merupakan id dari ambulans
string	status	Nilai dari status merupakan status dari ambulans
string	latitude	Nilai latitude merupakan koordinat garis lintang dari lokasi ambulans
string	longitude	Nilai longitude merupakan koordinat garis bujur dari lokasi ambulans

Pada Gambar 4.3 menggambarkan proses yang terjadi pada node *publisher* ketika mengirimkan data ambulans. *Publisher* menginstansiasi variabel *client*, dan menginisialisasi variable stop, id dan status. Setelah itu *publisher* melakukan perulangan selama nilai stop sama dengan *false*. Kemudian *publisher* melakukan pengecekan apakah *publisher* telah terhubung dengan *broker*. Jika belum maka akan menghubungkan *publisher* dengan *broker*, jika sudah maka akan *publish* data.

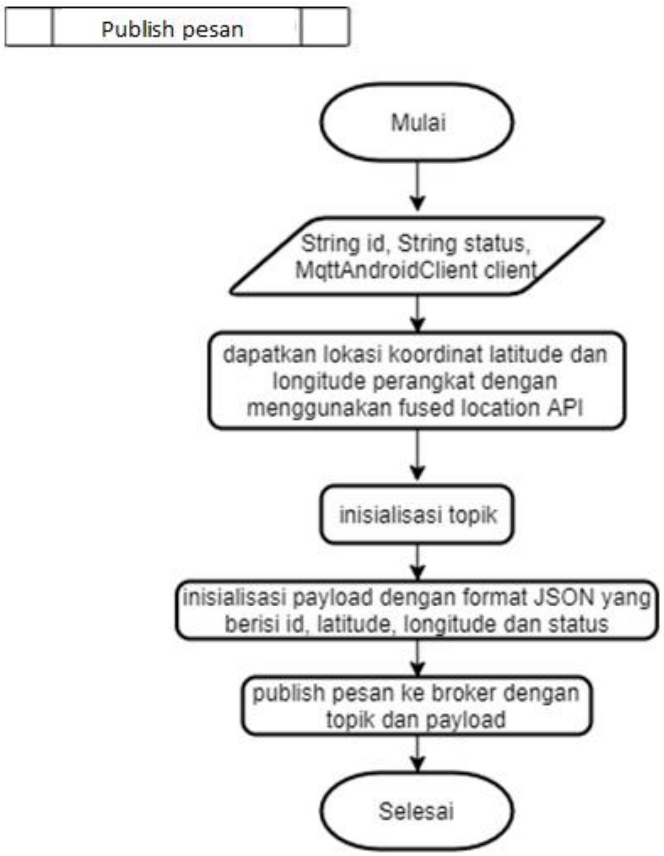


Gambar 4. 3 Diagram Alir *Publish* pada Node *Publisher*

Pada Gambar 4.4 menggambarkan proses untuk menghubungkan *publisher* dengan *broker* dan Gambar 4.5 menggambarkan proses *publish*. Pada proses *publish* terlebih dahulu mendapatkan lokasi dalam bentuk koordinat melalui GPS dengan fused location API sebelum melakukan *publish* ke *broker*.

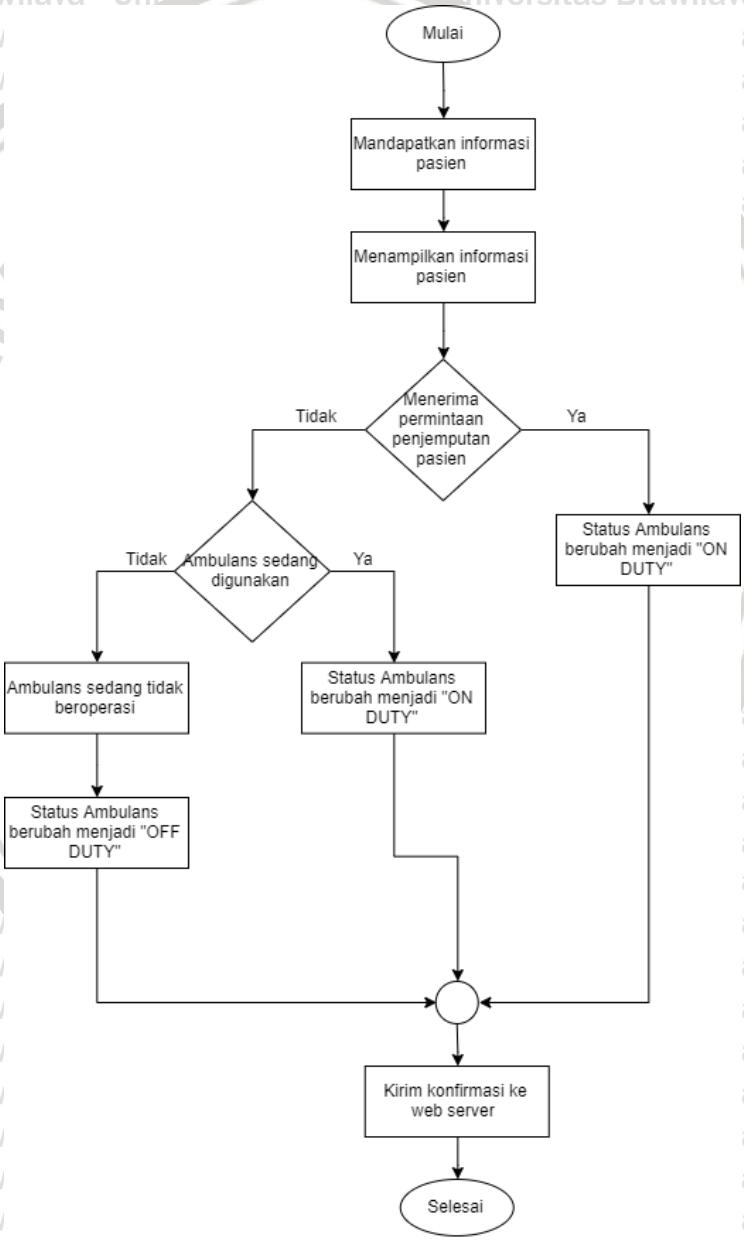


Gambar 4. 4 Diagram Alir Menghubungkan *Publisher* dan *Subscribe*



Gambar 4. 5 Diagram Alir *Publish*

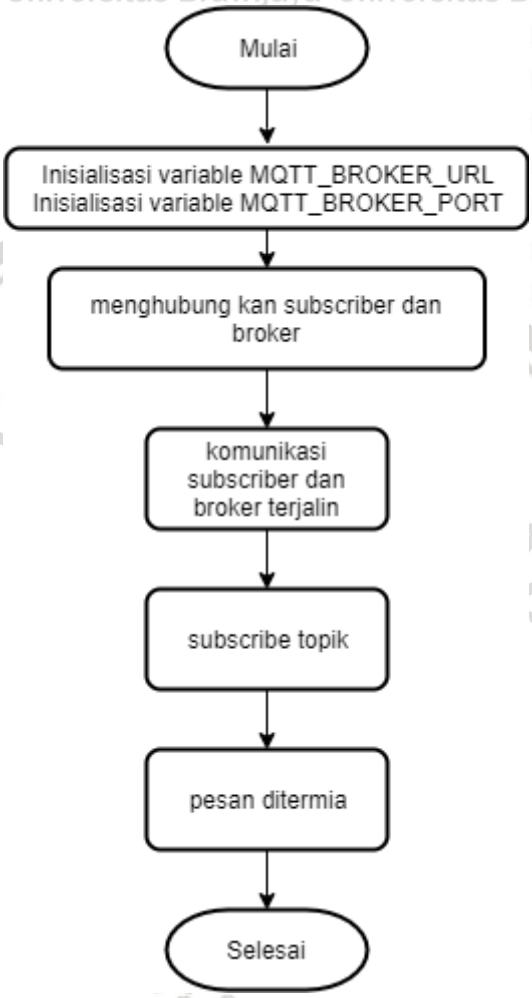
Pada Gambar 4.6 menggambarkan proses ketika menerima informasi pasein dari web server. Web server mengirimkan informasi pasien agar ambulans dapat menjemput pasien tersebut. Ketika menerima informasi pasien, pengguna akan memilih untuk menerima atau menolak permintaan penjemputan pasien. Jika menerima, makan akan mengubah status ambulans menjadi “ON DUTY” karena sekarang ambulans akan menjemput pasien. Jika menolak, maka pengguna akan memilih alasan menolak, jika menolak penjemputan pasien karena ambulans sedang digunakan, maka status ambulans berubah menjadi “ON DUTY”, jika menolak penjemputan pasien karena ambulans sedang tidak beroperasi, maka akan mengubah status menjadi “OFF DUTY”. Terakhir, akan mengirimkan konfirmasi ke web server.



Gambar 4. 6 Diagram Alir Menerima Informasi Pasien

4.2.3 Perancangan Node *Subscriber*

Node *subscriber* merupakan node web server yang menerima data ambulans dan melakukan layanan penentuan ambulans untuk pasien. Pada *subscriber*, data yang diterima akan disimpan ke basis data. Gambar 4.7 akan menunjukan diagram alir *subsciber* melakukan *subscribe* dan menerima data.



Gambar 4. 7 Diagram Alir *Subscribe*

Pada Gambar 4.7 dijelaskan mengenai diagram alir proses melakukan *subscribe*. *Subsciber* akan mencoba terhubung dengan *broker*, ketika *subscriber* dan *broker* berhasil terhubung berarti komunikasi antara *subscriber* dan *broker* terjalin. Setelah itu akan melakukan *subscribe* ke *broker*. Setelah melakukan *subscribe*, *subscriber* menunggu menerima data yang diteruskan oleh *broker*.

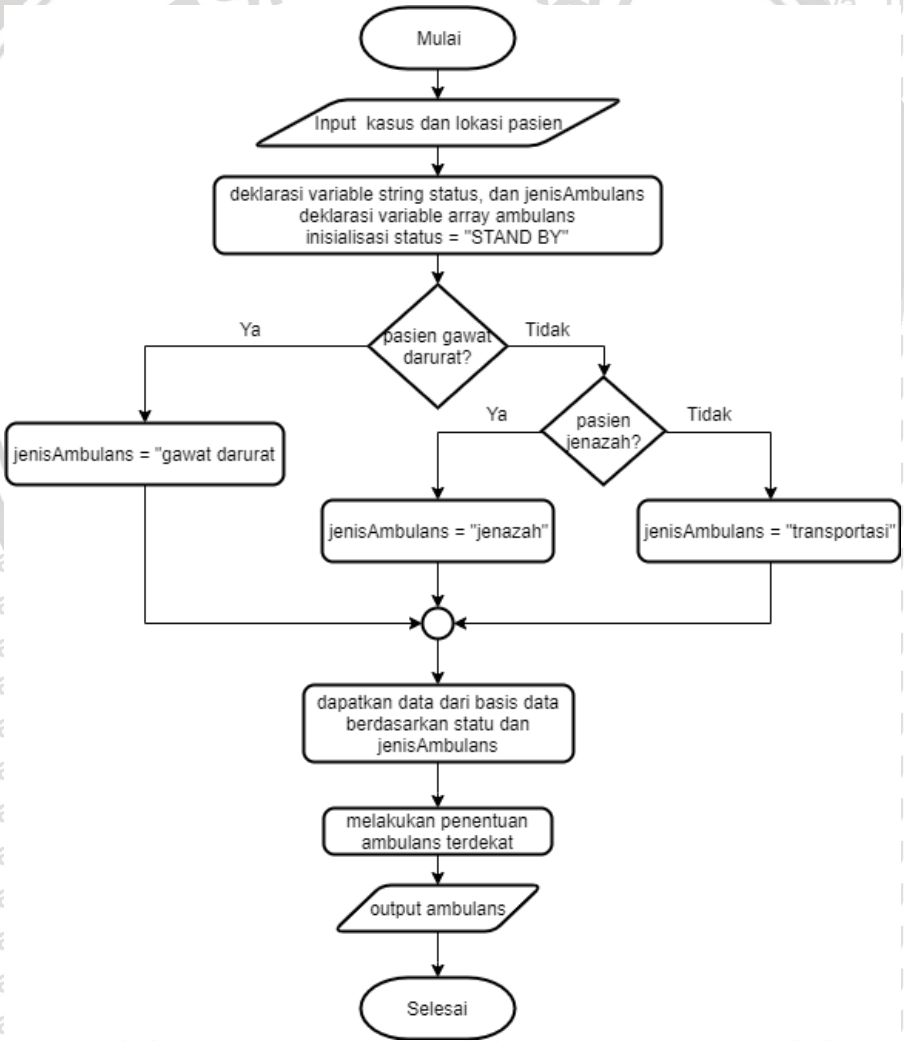
Node *subscriber* sebagai web server memiliki fungsi untuk menerima permintaan pengiriman ambulans oleh aplikasi perangkat bergerak pasien. Ambulans yang dikirim akan ditentukan berdasarkan lokasi pasien, status ambulans dan kasus pasien. Kasus pasien diperlukan untuk menentukan jenis

ambulans yang dibutuhkan oleh pasien. Tabel 4.4 merupakan kategori kasus pasien.

Tabel 4. 4 Kasus Pasien

Kegawatdaruratan	Kasus	Jenis Ambulans
Gawat darurat	Keadaan pasien gawat darurat	Ambulans gawat darurat
Tidak gawat darurat	Keadaan pasien tidak gawat darurat	Ambulans transportasi
	Jenazah	Ambulans jenazah

Setelah menyeleksi ambulans yang tersedia dengan kasus yang sesuai dengan pasien, akan dilakukan perhitungan jarak terdekat. Gambar 4.8 berikut menjelaskan cara kerja penentuan ambulans



Gambar 4. 8 Diagram Alir Penentuan Ambulans

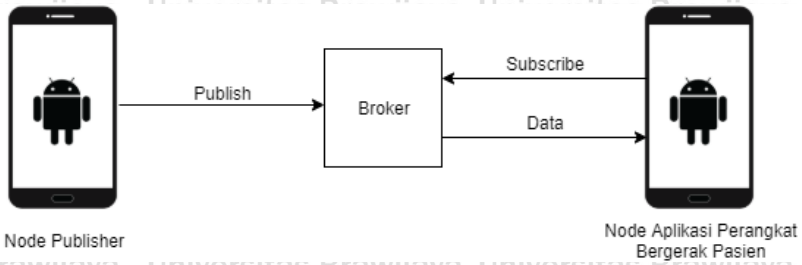
4.2.4 Perancangan Node Aplikasi Perangkat Bergerak Pasien

Node aplikasi perangkat bergerak pasien merupakan aplikasi android yang digunakan oleh pasien untuk mendapatkan ambulans. Aplikasi ini akan melakukan HTTP request ke Node *Subscriber* yang merupakan web server untuk mendapatkan ambulans yang dibutuhkan berdasarkan kasus dan lokasi pasien. Aplikasi ini akan memanfaatkan Google Play Service Location API untuk mendapatkan lokasi perangkat dalam bentuk koordinat. Pada aplikasi ini akan terdapat *form* untuk memilih kasus pasien yang membutuhkan ambulans. Data yang dikirimkan merupakan titik koordinat garis bujur lokasi pasien, titik koordinat garis lintang lokasi pasien dan kasus pasien. Titik koordinat garis lintang dan titik koordinat garis bujur untuk menentukan lokasi pasien. Titik koordinat didapatkan dengan menggunakan GPS. Kasus pasien merupakan keadaan dari pasien yang membutuhkan ambulans. Kasus pasien dibutuhkan untuk mengetahui jenis ambulans yang dibutuhkan oleh pasien. Tabel 4.5 menunjukan data pasien ketika melakukan POST request ke web server.

Tabel 4. 5 POST *Request* Data Pasien

Tipe Data	Key	Value
string	username	Nilai dari username merupakan string unik yang berbeda antar pengguna
string	kasus	Nilai dari kasus pasien
string	latitude	Nilai latitude merupakan koordinat garis lintang dari lokasi ambulans
string	longitude	Nilai longitude merupakan koordinat garis bujur dari lokasi ambulans

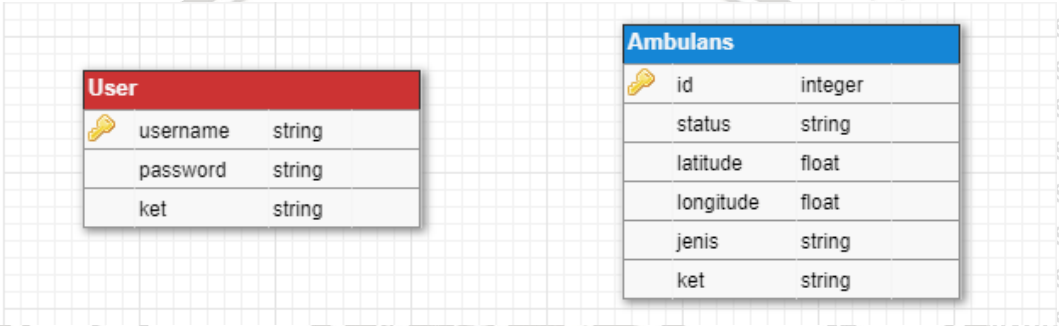
Aplikasi perangkat bergerak pasien yang melakukan pemesanan ambulans akan menampilkan *maps* dan menunjukan lokasi ambulans yang dipesan. Untuk mengetahui lokasi ambulans, aplikasi perangkat bergerak pasien akan mengimplementasikan metode komunikasi publish/subscribe. Aplikasi perangkat bergerak pasien akan berperan sebagai subscriber yang melakukan subscribe terhadap ambulans yang dipesan. Gambar 4.9 merupakan gambaran aplikasi perangkat bergerak pasien melakukan subscribe.



Gambar 4. 9 Aplikasi Pasien Sebagai Subscriber

4.2.5 Perancangan Basis Data

Perancangan basis data ini menjelaskan isi dan pengaturan data yang dibutuhkan. Gambar 4.10 akan menunjukan perancangan basis data.



Gambar 4. 10 Perancangan Basis Data

Pada Gambar 4.10 terdapat perancangan basis data *User* dan Ambulans. Tabel *User* berisi *username*, *password* dan ket yang semuanya bertipe data string. *Username* digunakan untuk mengidentifikasi pengguna. Password yang berarti kata sandi untuk memverifikasi identitas pengguna terhadap sistem. Ket merupakan keterangan dari user untuk menjelaskan apakah user merupakan pasien atau petugas ambulans.

Pada tabel ambulans berisi id, status, *latitude*, *longitude* dan jenis. Id digunakan untuk mengidentifikasi ambulans. Status merupakan status ambulans yaitu on duty, off duty atau stand by. *Latitude* dan *longitude* merupakan titik koordinat dari ambulans. Jenis merupakan jenis ambulans yaitu gawat darurat, transportasi dan emergency. Keterangan untuk menjelaskan rumah sakit atau puskesmas atau yayasan ambulans apa yang menaungi ambulans tersebut

4.2.6 Perancangan Pengujian

Perancangan pengujian dilakukan agar dapat menguji sistem sesuai dengan kebutuhan yang diperlukan. Pada perancangan pengujian ini terdapat perancangan pengujian fungsional, perancangan pengujian akurasi, dan perancangan pengujian *response time*.

4.2.6.1 Perancangan Pengujian Fungsional

Perancangan pengujian fungsional dilakukan untuk memastikan sistem memenuhi kebutuhan yang dibutuhkan. Kebutuhan didapatkan dari kebutuhan fungsional yang telah didefinisikan. Tabel 4.6 berikut merupakan skenario pengujian kebutuhan fungsional.

Tabel 4. 6 Skenario Pengujian Kebutuhan Fungsional

Kode Fungsi	Kebutuhan Fungsional	Skenario Pengujian
KF-01-01	Node *publisher* harus dapat menentukan lokasi dari perangkat node menggunakan GPS	1. Menyalakan location service pada perangkat node publisher 2. Node *publisher* mengimplementasikan Google Play Service Location API untuk mendapatkan lokasi perangkat dalam bentuk koordinat 3. Menampilkan lokasi yang didapatkan berupa koordinat pada halaman utama
KF-01-02	Node *publisher* harus dapat melakukan *publish* data secara terus menerus ke broker dengan format JSON	1. Node *publisher* menghubungkan *publisher* dengan *broker* 2. Node *publisher* melakukan *publish* 3. Menampilkan data yang dikirimkan pada *subscriber*
KF-01-03	Node publisher harus dapat menerima informasi pasien yang membutuhkan ambulans	1. Node *subscriber* sebagai web server mengirimkan informasi pasein ke ambulans 2. Node *publisher* menampilkan data yang dikirimkan node *subscriber*
KF-01-04	Node publisher harus dapat menolak atau manerima melakukan penjemputan pasien	1. Node *subscriber* sebagai web server mengirimkan informasi pasein ke ambulans 2. Node publisher menampilkan informasi pasien 3. Node publisher menampilkan tombol untuk menolak atau menerima penjemputan pasien 4. Node *publisher* mengirimkan konfirmasi menolak atau menerima penjemputan pasien ke node *subscriber* yang merupakan web server 5. Node *subscriber* sebagai web

		server menampilkan konfirmasi dari node *publisher*
KF-02-01	Node *subscriber* harus dapat melakukan *subscribe* ke broker	1. Node *subscriber* menghubungkan *subscriber* dengan broker 2. Menampilkan bahwa berhasil melakukan *subscribe* 3. Node *subscriber* menerima data sesuai dengan topik
KF-02-02	Node *subscriber* harus dapat menerima data dari *publisher* yang diteruskan oleh broker	1. Node *subscriber* menghubungkan *subscriber* dengan broker 2. Menampilkan data yang diterima ketika menerima data
KF-02-03	Node *subscriber* harus dapat melakukan pembaruan data yang diterima dari *publisher* pada basis data	1. Menjalankan node *subscriber* dan melakukan *subscribe* 2. Menjalankan *broker* 3. Menjalankan program *publisher* untuk mengirimkan data 4. Node *subscriber* menerima data yang diteruskan oleh *broker* dari *publisher* 5. Node *subscriber* melakukan pembaruan pada basisdata berdasarkan data yang diterima 6. Menampilkan data yang diperbarui
KF-02-04	Node *subscriber* harus dapat melakukan penentuan ambulans yang dibutuhkan oleh pasien berdasarkan status ambulans, kasus pasien, dan lokasi ambulans	1. Node *subscriber* menerima input lokasi pasien, dan kasus pasien 2. Node *subscriber* menjalankan fungsi untuk menentukan jenis ambulans berdasarkan status ambulans dan jarak ambulans 3. Menampilkan hasil penentuan ambulans
KF-02-05	Node *subscriber* dapat menerima *request* dari node aplikasi perangkat bergerak pasien untuk memberikan *response* berupa ambulans yang sesuai dengan lokasi pasien dan kasus pasien	1. Node *subscriber* menerima request dari node aplikasi perangkat bergerak pasien 2. Menampilkan *request* yang diterima dan *response* yang dikirimkan
KF-02-06	Node *subscriber* dapat mengirimkan informasi pasien ke node *publisher*	1. Node *subscriber* menerima request dari node aplikasi perangkat bergerak pasien

		2. Node *subscriber* menentukan ambulans untuk pasien 3. Node *subscriber* mengirimkan informasi pasein ke node *publisher* dengan publisher yang memiliki ID ambulans yang sama dengan ID ambulans yang telah ditentukan sebelumnya 4. Node *publisher* menampilkan infromasi yang diterima oleh node *subscriber*
KF-03-01	Node aplikasi perangkat bergerak pasien dapat mengirimkan *request* ke node *subscriber* untuk mendapatkan ambulans yang sesuai dengan lokasi pasien dan kasus pasien	1. Menyalakan location service pada perangkat 2. Node aplikasi perangkat bergerak pasein mengimplementasikan Google Play Service Location API untuk mendapatkan lokasi perangkat dalam bentuk koordinat 3. Node aplikasi pasien menampilkan form untuk menerima input dari user 4. Node aplikasi pasien mengirimkan *request* ke *subscriber*/webserver 5. Menampilkan *request* yang dikirimkan dan *response* yang diterima

4.2.6.2 Pengujian Penentuan Ambulans

Pengujian penentuan ambulans dilakukan untuk mengetahui apakah penentuan ambulans menghasilkan hasil yang benar. Penentuan ambulans dilakukan dengan menentukan jenis ambulans berdasarkan kasus pasien. Kasus pasien sakit gawat darurat akan mendapatkan ambulans dengan jenis gawat darurat, kasus pasien sakit tidak gawat darurat akan mendapatkan ambulans dengan jenis transportasi dan pasien jenazah akan mendapatkan ambulans jenazah. Setelah diketahui jenis ambulans, akan mengambil data ambulans pada basis data berdasarkan jenis yang telah diketahui dan status STAND BY. Dari data yang didapatkan pada basis data, diketahui titik koordinat ambulans yang dapat digunakan menghitung jarak antar ambulans dan pasien. Dengan diketahuinya jarak antar ambulans dan pasien dapat dilakukan perbandingan jarak untuk menemukan ambulans dengan jarak terdekat.

Ambulans terdapat 3 jenis yaitu ambulans gawat darurat, ambulans transportasi dan ambulans jenazah. Oleh karena itu pada pengujian penentuan jenis ambulans, dilakukan pengujian untuk masing-masing jenis ambulans.

4.2.6.3 Perancangan Pengujian Akurasi

Pengujian Akurasi Sistem dilakukan untuk mengetahui tingkat akurasi implementasi formula haversine dalam menentukan jarak antara titik koordinat perangkat node *publisher* dan titik koordinat node aplikasi perangkat bergerak pasien. Nilai akurasi didapatkan dari perhitungan persentase galat (*error*) relatif. Persentase galat relatif ditentukan dari perbandingan antara galat absolut dan nilai sebenarnya seperti pada persamaan 4.1.

(4.1)

dimana:

Er : Galat Relatif

Ea : Galat Absolut

x : Nilai Sejati

Galat absolut adalah nilai mutlak dari selisih antara nilai sebenarnya dengan nilai pendekatan. Dalam penelitian ini nilai sebenarnya didapatkan dengan menggunakan Google Maps JavaScript API untuk menentukan jarak antara 2 titik koordinat. Nilai pendekatan merupakan nilai yang dihasilkan dengan menggunakan formula haversine untuk menentukan jarak antara 2 titik koordinat. Galat absolut ditunjukan pada persamaan 4.2.

(4.2)

dimana:

Ea : Galat Absolut

x : Nilai Sejati

x1 : Nilai pendekatan

4.2.6.4 Perancangan Pengujian *Response Time*

Pengujian *Response Time* adalah pengujian waktu respons yang mengacu pada waktu yang diperlukan untuk suatu node sistem untuk menanggapi permintaan. Pengujian *response time* pada penelitian dibagi menjadi dua pengujian.

Pengujian *response time* yang pertama adalah pengujian waktu yang diperlukan untuk mengirimkan data lokasi dari *publisher* ke *subscriber*. Waktu

dihitung dari penentuan lokasi pada node *publisher* sampai data sampai pada *subscriber*. Pengujian *response time* ini dilakukan sebanyak 10 kali percobaan.

Pengujian *response time* yang kedua adalah *response time* dari aplikasi pasien ke node subscriber yang merupakan web server dalam menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans pada basis data. Variasi jumlah data pada basis data adalah 1000, 2000, 3000, 4000 dan 5000 data ambulans. Variasi pada jumlah data ambulans dilakukan karena pada sistem ini terjadi perulangan untuk menghitung satu persatu jarak ambulans dengan pasien. Pengujian *response time* dilakukan dengan menggunakan alat bantu aplikasi postman.

BAB 5 IMPLEMENTASI SISTEM

Implementasi adalah tahap penerapan sisten yang telah dirancang. Dalam penelitian ini, metode komunikasi *publish*/*subscribe* diimplementasikan menggunakan protokol MQTT pada aplikasi perangkat bergerak android dan web server. Aplikasi perangkat bergerak android merupakan node publisher yang mengirimkan data ambulans, yaitu status ambulans dan titik koordinat ambulans. Web server merupakan node *subscriber* yang menerima data ambulans. Data ambulans yang didapatkan akan disimpan dalam basis data dan akan digunakan ketika menjalankan layanan penentuan ambulans yang sesuai dengan kebutuhan pengguna pasien.

5.1 Implementasi Node *Publisher*

Node *publisher* merupakan aplikasi android yang digunakan oleh petugas ambulans untuk mengirimkan titik koordinat dan status ambulans. Titik koordinat didapatkan dengan menyalakan layanan lokasi pada ponsel dan status ambulans didapatkan dari inputan user. Data ambulans dikirimkan dengan menggunakan protokol MQTT. Untuk mengimplementasikan protokol MQTT pada aplikasi android, digunakan Paho Android Service yang merupakan library klien MQTT.

5.1.1 Implementasi Mendapatkan Lokasi

Lokasi didapatkan menggunakan GPS melalui FusedLocationProvider yang dibawah Google API. FusedLocationProvider digunakan untuk berinteraksi dengan posisi menggunakan fused location provider. Penentuan kualitas layanan untuk pembaruan lokasi dari FusedLocationProvider digunakan objek LocationRequest. Untuk mendapatkan lokasi terbaru digunakan LocationCallback. Pada Tabel 5.1 menunjukan baris kode untuk mendefinisikan objek FusedLocationProvider dan LocationRequest. Pada baris 6-8 Tabel 5.1 mengatur memanggil method untuk mengatur prioritas dan interval waktu. Pada baris ke 9 Tabel 5.1 memanggil method callback untuk memperbarui lokasi. Lokasi yang didapatkan akan diinisialisasi ke variable locationAmbulance.

Tabel 5. 1 Kode Program Mendefinisikan Pengaturan Lokasi

Fungsi onCreate

```
1   protected void onCreate(Bundle savedInstanceState) {
2   . . .
3
4       mFusedLocationClient                                    =
LocationServices.getFusedLocationProviderClient(this);
5       locationRequest = LocationRequest.create();
6       locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURAC
    Y);
7       locationRequest.setInterval(UPDATE_INTERVAL);
8       locationRequest.setFastestInterval(FASTEST_INTERVAL);
9       locationCallback = new LocationCallback() {
10          @Override
```

```
11          public void onLocationResult(LocationResult locationResult) {
12              if (locationResult == null) {
13                  return;
14              }
15              for (Location location : locationResult.getLocations()) {
16                  if (location != null) {
17                      locationAmbulance=location;
18                  }
19              }
20          }
21      };
22  }
```

Method requestLocation yang ditunjukan pada baris ke 2 Tabel 5.2 dipanggil, yang meneruskan objek LocationRequest dan LocationCallback. Method ini untuk mendapatkan update dengan menggunakan pendekatan callback.

Tabel 5. 2 Kode Program Memperbarui Lokasi

```
Fungsi Memperbarui Lokasi
1   public void startLocationUpdate(){
2       mFusedLocationClient.requestLocationUpdates(locationRequest,locat
    ionCallback, Looper.getMainLooper());
3   }
```

5.1.2 Implementasi *Event Handler* Memulai dan Menghentikan *Publish*

Tabel 5.3 merupakan potongan baris kode program yang dipanggil ketika menekan tombol "Pilih". Tombol "Pilih" digunakan untuk melakukan *publish* secara terus menerus ke *broker*. Untuk melakukan *publish* data ke *broker*, akan dibuat *background thread*. *Background thread* adalah *thread* yang berjalan dibalik layar ketika *thread* utama berjalan. Ketika fungsi ini dipanggil akan menginisialisasi variabel stop menjadi *false*. Variabel ini digunakan untuk memulai dan menghentikan *publish* data ke broker. Pada baris 4 dilakukan seleksi kondisi untuk mengetahui apakah *thread* telah diakhiri atau kosong. Jika *thread* telah diakhiri atau kosong, maka akan membuat *thread* seperti yang ditunjukan oleh baris 34. Sebelum dibuat *thread*, terlebih dahulu menginstansiasi *interface Runnable* yang ditunjukan pada baris 5-33. *Interface Runnable* merupakan *template* untuk objek, dimana objek tersebut ditujukan untuk di eksekusi oleh *thread*. *Interface Runnable* mendefinisikan metod run(), berisi kode yang akan di eksekusi oleh *thread*. Pada baris 8-31 merupakan kode yang akan dijalankan oleh *thread*. Kode tersebut merupakan perulangan untuk melakukan *publish* dengan memanggil fungsi publish(id, status, client) setiap 1 detik. Sebelum melakukan *publish*, dilakukan seleksi kondisi terlebih dahulu untuk mengetahui apakah *publisher* sudah terhubung dengan *broker*, jika belum maka akan memanggil method connectToMqtt(client) untuk menghubungkan *publisher* dengan *broker*.

Tabel 5. 3 Implementasi *Event Handler* Memulai *Publish*

```
Fungsi Memulai Publish
```

```
1  public void btnPilih_onClick(View view){
2      . . .
3      stop= false;
4      if(bgthread==null||bgthread.getState()==
   Thread.State.TERMINATED){
5          Runnable runnable = new Runnable() {
6              @Override
7              public void run() {
8                  try{
9                      while(!stop){
10                         handler.post(new Runnable() {
11                             public void run() {
12                                 if(client.isConnected()){
13                                     if(locationAmbulance!=null){
14                                         publish(id,status, client);
15                                         loadingMain.setVisibility(Vie
   w.GONE);
16                                     }
17                                     else {
18                                         loadingMain.setVisibility(View
   .VISIBLE);
19                                     }
20                                 }
21                                 else{
22                                     connectToMqtt(client);
23                                 }
24                             }
25                         });
26                         Thread.sleep(1000);
27                     }
28                 }
29                 catch (InterruptedException e){
30                     e.printStackTrace();
31                 }
32             }
33         };
34         bgthread = new Thread(runnable);
35         bgthread.start();
36     }
37 }
```

Tabel 5.4 merupakan potongan baris kode program yang dipanggil ketika menekan tombol "Stop". Tombol "Stop" digunakan untuk menghentikan melakukan *publish* ke *broker* dengan mengubah nilai stop menjadi *true*.

Tabel 5. 4 Implementasi *Event Handler* Menghentikan *Publish*

Fungsi Menghentikan Publish

```
1  public void btnStop_onClick(View view){
2      . . .
3
4      stop = true;
5  }
```

5.1.3 Implementasi Menghubungkan *Publisher* dengan *Broker*

Paho Android Service adalah antar muka ke *library* Paho Java MQTT Client untuk android. Koneksi MQTT di enkapsulasi pada Android-Service yang berjalan pada background aplikasi android agar tetap berjalan ketika aplikasi android

beralih dari aktivitas satu ke aktivitas lain. Untuk itu layanan Paho Android Service perlu di deklarasikan pada AndroidManifest.xml seperti pada Tabel 5.5.

Tabel 5. 5 Deklarasi layanan MQTT Android pada Aplikasi

Deklarasi layanan MQTT

```
1  <manifest xmlns:android=http://schemas.android.com/apk/res/android
   package="com.example.ambulans">
2      . . .
3     <application . . .>
4         . . .
5          <service
           android:name="org.eclipse.paho.android.service.MqttService">
6          </service>
7
8     </application>
9  </manifest>
```

Tabel 5.6 merupakan potongan baris kode program untuk membuat koneksi MQTT. Pada baris ke 9 fungsi generateClientId digunakan untuk menghasilkan id pengguna secara acak. Pada baris ke 10 instansiasi MQTT Android Client, yang akan bind ke Paho Android Service. Selanjutnya pada baris ke 12 akan memanggil method connectToMqtt untuk menghubungkan *publisher* dengan *broker*.

Tabel 5. 6 Implementasi *Bind* ke *Paho Android Service*

Bind ke Paho Android Service

```
1  . . .
2
3  MqttAndroidClient client;
4
5  @Override
6  protected void onCreate(Bundle savedInstanceState) {
7      . . .
8
9      String clientId = MqttClient.generateClientId();
10     client   =   new   MqttAndroidClient(this.getApplicationContext(),
       "tcp://192.168.1.18:1883", clientId);
11
12     connectToMqtt(client);
13 }
14
15 . . .
```

Tabel 5.7 merupakan potongan baris kode program untuk menghubungkan *publisher* dengan *broker*. Pada baris ke 3 dipanggil method connect dari MqttAndroidClient. Dengan memanggil method tersebut, klien akan mencoba tersambung ke *broker* dan mengembalikan token. Token tersebut digunakan untuk mendapatkan pemberitahuan ketika koneksi MQTT terhubung atau terjadi kesalahan. Pada kode tersebut akan mencetak "Success" ketika terhubung dan mencetak "Failed" ke konsol bila terjadi kesalahan.

Tabel 5. 7 Impementasi Fungsi untuk Menghubugkan *Publisher* dengan *Broker*

Fungsi Menghubungkan Publisher dengan Broker

```
1  public void connectToMqtt(final MqttAndroidClient client) {
2      try {
3          IMqttToken token = client.connect();
```

```
4           Log.d("connectToMQTT", "try");
5           token.setActionCallback(new IMqttActionListener() {
6               @Override
7               public void onSuccess(IMqttToken asyncActionToken) {\
8                   Log.d("connectToMQTT","success");
9               }
10
11              @Override
12
13              public    void    onFailure(IMqttToken    asyncActionToken,
    Throwable exception) {
14                  Log.d("connectToMQTT","failed");
15
16              }
17          });
18      } catch (MqttException e) {
19          Log.d("connectToMQTT", "catch");
20          e.printStackTrace();
21      }
22  }
```

5.1.4 Implementasi *Publish*

Pada Tabel 5.8 merupakan potongan baris kode program untuk *publish*. Sebelum melakukan *publish*, terlebih dahulu mencari lokasi saat ini dengan memanggil variabel locationAmbulance. Setelah sukses mendapatkan lokasi perangkat, menginisialisasi variabel payload dengan format JSON yang berisi ID, titik koordinat latitude, titik koordinat longitude dan status. Dan menginisialisasi topik. Pada baris 17 mengubah String ke UTF-8 bytes dan dilakukan publish berdasarkan topik.

Tabel 5. 8 Implementasi Fungsi untuk *Publish*

Fungsi Publish

```
1   private void publish(final String ID, final String status, final
    MqttAndroidClient client){
2
3       String latitude =String.valueOf(locationAmbulance.getLatitude());
4       String                                                  longitude
    =String.valueOf(locationAmbulance.getLongitude());
5
6       . . .
7
8       String payload;
9       payload = "{" +
                  "\"id\":"+ID+"," +
                  "\"latitude\":\""+latitude+"\"," +
                  "\"longitude\":\""+longitude+"\"," +
                  "\"status\":\""+status+"\"}";
10      String topic = "TestingSkripsiPutri/"+ID;
11      byte[] encodedPayload = new byte[0];
12      try {
13          encodedPayload = payload.getBytes("UTF-8");
14          MqttMessage message = new MqttMessage(encodedPayload);
15          client.publish(topic, message);
16      } catch (UnsupportedEncodingException | MqttException e) {
17          e.printStackTrace();
18      }
19  }
```

5.1.5 Implementasi Menerima Informasi Pasien

Node *publisher* sebagai aplikasi perangkat bergerak untuk ambulans dapat menerima informasi permintaan penjemputan pasien dari node subscriber yang merupakan web server. Node *publisher* kemudian melakukan konfirmasi menerima atau menolak permintaan penjemputan pasien.

Informasi permintaan penjemputan pasien dikirimkan oleh web server menggunakan Firebase Cloud Messaging (FCM). Dengan menggunakan Firebase server dapat mengirimkan data. Firebase Cloud Messaging memiliki dua jenis pesan yang dapat dikirimkan dari server ke klien, yaitu notification message dan data message. Jenis pesan data message digunakan ketika server perlu mengirimkan data ke klien dan aplikasi klien yang bertanggung jawab untuk memproses data tersebut. Pada node publisher, untuk menerima data informasi pasien digunakan jenis pesan data message. pada Tabel 5.9 merupakan implementasi menerima pesan dari webserver. Pada baris ke 9-14, Data diterima dalam format JSON akan diinisialisasi ke dalam variable dan akan dikirimkan ke activity untuk ditampilkan. Untuk mengirimkan ke activity, digunakan broadcastmanager yang dapat dilihat pada baris ke 17-25.

Tabel 5. 9 Implementasi Menerima Pesan dari Webserver

Fungsi Menerima Pesan dari Webserver

```
1   @Override
2   public void onMessageReceived(RemoteMessage remoteMessage) {
3       Log.d(TAG, "From: " + remoteMessage.getFrom());
4
5       if (remoteMessage.getData().size() > 0) {
6           Log.d(TAG, "Message data payload: "+remoteMessage.getData());
7           try {
8               JSONObject jsonObject = new
    JSONObject(remoteMessage.getData());
9               String nama = jsonObject.getString("nama");
10              String alamat = jsonObject.getString("alamat");
11              String noPonsel = jsonObject.getString("noPonsel");
12              String tokenPasien = jsonObject.getString("tokenPasien");
13              6WULQJ'DW3DVLHQ0VRQ2EMHFW'HW6WULQJ'DW3DVLHQ
14              String lRQ3DVLHQ0VRQ2EMHFW'HW6WULQJ'RQ3DVLHQ
15
16              Context context=this;
17              Intent intent = new Intent("TERIMA-PASIEN");
18              intent.putExtra("nama",nama);
19              intent.putExtra("alamat",alamat);
20              intent.putExtra("noPonsel",noPonsel);
21              intent.putExtra("tokenPasien",tokenPasien);
22              intent.putExtra('DW3DVLHQ'DW3DVLHQ
23              intent.putExtra('RQ3DVLHQ'RQ3DVLHQ
24
25              LocalBroadcastManager.getInstance(context).sendBroadcast(
    intent);
26          }
27          catch (JSONException e){
28
29          }
30      . .
31
32      }
33
```

```
34  }
```

Pada kode program Tabel 5.10 data yang diterima akan diisialisasi pada variable dan memanggil fungshi ShowPopupTerimaPasein untuk menampilkan data informasi pasien.

Tabel 5. 10 Implementasi Menerima Data Pasien untuk Ditampilkan

```
Kode Menerima Data untuk Ditampilkan
1   private BroadcastReceiver mMessageReceiver = new BroadcastReceiver()
2   {
3       @Override
4       public void onReceive(Context context, Intent intent) {
5           namaPasien = intent.getStringExtra("nama");
6           alamatPasien = intent.getStringExtra("alamat");
7           noPonselPasien = intent.getStringExtra("noPonsel");
8           tokenPasien = intent.getStringExtra("tokenPasien");
9           String id =  mPreferences.getString("id",null);
10          ODW3DVLHQ EQWHQW HW6WULQJ( VWUD DW3DVLHQ
11          ORQ3DVLHQ EQWHQW HW6WULQJ( VWUD RQ3DVLHQ
12
13          ShowPopupTerimaPasien(namaPasien,          noPonselPasien,
    alamatPasien, tokenPasien, id, latPasien, lonPasien);
14      }
15  };
```

Pada kode program Tabel 5.11 akan ditampilkan data pasien dengan menampilkan dialog popup. Pada dialog popup terdapat tombol agar pengguna dapat menerima atau menolak untuk ambulans menjemput pasien. Pada baris 5-14 merupakan kode ketika pengguna menerima menjemput pasien. Jika menerima, pada baris ke 8 dipanggil method kirimKonfirmasi. Method kirimkonfirmasi akan mengirimkan ke server bahwa ambulans akan menjemput pasien. Pada baris 9-10, akan mengubah status menjadi “ON DUTY”. Setelah status berubah, akan memanggil method doSomething untuk membuka halaman baru yang menampilkan *maps* dan data pasien. Jika menolak, pada baris ke 21, akan menampilkan dialog popup untuk memilih alasan menolak menjemput pasien.

Tabel 5. 11 Implementasi Menampilkan Data Pasien

```
Fungsi Menampilkan data Pasien
1   public  void  ShowPopupTerimaPasien(String  nama,  String  noponsel,
2   String alamat, final String tokenPasien, final String id){
3       . . .
4
5   btnterima.setOnClickListener(new View.OnClickListener() {
6           @Override
7           public void onClick(View view) {
8               kirimKonfirmasi(tokenPasien, "ya", id);
9               spinner.setSelection(1);
10              status = spinner.getSelectedItem().toString();
11              doSomething(nama,  noponsel,  alamat,  tokenPasien,  id,
    latPsn,lonPsn);
12
13              . . .
14          }
15      });
16
```

```
17      btntolak.setOnClickListener(new View.OnClickListener() {
18          @Override
19          public void onClick(View view) {
20              . . .
21              ShowPopupMenolak(tokenPasien,id);
22          }
23      });
24
25      . . .
}
```

Pada kode program pada Tabel 5.12 merupakan fungsi untuk membuka halaman baru. Halaman tersebut dibuka ketika pengguna aplikasi ambulans menerima pesanan menjemput pasien. Pada halaman itu akan ditampilkan maps dengan menampilkan lokasi titik pasien dan titik ambulans serta menampilkan data pasien seperti nomor ponsel, alamat lokasi pasien dan nama pasien. Pada baris ke 3 akan dibuat objek intent. Intent berfungsi untuk berpindah halaman. Selanjutnya pada baris 4-10 akan menggunakan method putExtra() untuk mengirimkan data ke halaman yang dituju. Pada baris 11 akan dimulai membuka halaman baru.

Tabel 5. 12 Implementasi Membuka Halaman Baru

```
Fungsi Membuka Halaman Baru
1   public void doSomething(String nama, String noponsel, String alamat,
2   String tokenPasien, String id, String latP, String lonP){
3       Intent intent = new Intent(MainActivity.this, MapActivity.class);
4       intent.putExtra("nama", nama);
5       intent.putExtra("noponsel", noponsel);
6       intent.putExtra("alamat", alamat);
7       intent.putExtra("id", id);
8       intent.putExtra("tokenPasien", tokenPasien);
9       intent.putExtra("latPasien", latP);
10      intent.putExtra("lonPasien", lonP);
11      startActivity(intent);
12      finish();
13  }
```

Pada kode program di Tabel 5.13 akan ditampilkan dialog popup untuk memilih alasan menolak permintan penjemputan pasien. Pada dialog popup tersebut menampilkan dua tombol. Tombol yang pertama dipilih ketika menolak permintaan karena ambulans sedang digunakan dan tombol kedua dipilih ketika ambulans sedang tidak beroperasi. Pada baris 7-10 merupakan pengaturan tombol menolak permintaan karena ambulans sedang digunakan dan pada baris 18-27 merupakan pengaturan tombol menolak permintaan karena ambulans sedang tidak beroperasi. Pada kedua tombol tersebut akan dipanggil fungsi kirimKonfirmasi untuk mengirimkan konfirmasi ke web server, dapat dilihat pada baris 10 dan baris 21.

Pada baris ke 11-12 tabel 5.13 , ketika memilih tombol menolak permintaan karena ambulans sedang tidak beroperasi, status akan berubah menjadi “OFF DUTY”. Setelah status berubah, pada baris ke 13 akan memanggil method update untuk mengirimkan status terbaru ambulans ke webserver.

Pada baris ke 22-23 tabel 5.13 , ketika memilih tombol menolak permintaan karena ambulans sedang digunakan, status akan berubah menjadi “ON DUTY”. Setelah status berubah, pada baris ke 24 akan memanggil method update untuk mengirimkan status terbaru ambulans ke webserver.

Tabel 5. 13 Implementasi Menampilkan Dialog Popup Menolak Permintaan

Fungsi Menampilkan Popup Menolak Permintaan

```
1  public void ShowPopupMenolak(final String tokenPasien, final String
2  id){
3      dialog_menolak.setContentView(R.layout.popup_menolak);
4      final Button btnOnduty = (Button)
   dialog_menolak.findViewById(R.id.btnOnduty_pm);
5      final Button btnOffduty = (Button)
   dialog_menolak.findViewById(R.id.btnOffduty_pm);
6
7      btnOffduty.setOnClickListener(new View.OnClickListener() {
8          @Override
9          public void onClick(View view) {
10             kirimKonfirmasi(tokenPasien, "OFF DUTY", id);
11             spinner.setSelection(2);
12             status = spinner.getSelectedItem().toString();
13             update();
14             dialog_menolak.dismiss();
15         }
16     });
17
18     btnOnduty.setOnClickListener(new View.OnClickListener() {
19         @Override
20         public void onClick(View view) {
21             kirimKonfirmasi(tokenPasien, "ON DUTY", id);
22             spinner.setSelection(1);
23             status = spinner.getSelectedItem().toString();
24             update();
25             dialog_menolak.dismiss();
26         }
27     });
28
29     dialog_menolak.getWindow().setBackgroundDrawable(new
   ColorDrawable(Color.TRANSPARENT));
30     dialog_menolak.show();
31 }
```

Kode program Tabel 5.14 akan mengirimkan konfirmasi ke web server. Data yang dikirimkan merupakan tokenPasien, id dan konfirmasi. Data tokenPasien untuk mengetahui pasien mana yang membutuhkan ambulans. Token id merupakan id ambulans. Konfirmasi merupakan data yang berisi keterangan apakah ambulans menerima melakukan penjemputan pasien.

Tabel 5. 14 Kirim Konfirmasi Ambulans

Fungsi Mengirim Konfirmasi

```
1  public void kirimKonfirmasi(final String tokenPasien, final String
   konfirmasi, final String id){
2      StringRequest stringRequest = new
   StringRequest(Request.Method.POST, URL_SEND_KONFIRMASI, new
   Response.Listener<String>() {
3          @Override
4          public void onResponse(String response) {
5              try{
6                  JSONObject jsonObject = new JSONObject(response);
```

```
7                   String success = jsonObject.getString("success");
8
9                   if(success.equals("1")){
10                      Log.d("onResponse","success");
11                  }
12                  else{
13                      Log.d("onResponse","failed");
14                  }
15              }
16              catch (JSONException e){
17                  e.printStackTrace();
18                  Log.d("onResponse","JSON Exception");
19              }
20          }
21      }, new Response.ErrorListener() {
22          @Override
23          public void onErrorResponse(VolleyError error) {
24              Log.d("onErrorResponse","VolleyError");
25          }
26      })
27      {
28          @Override
29          protected Map<String, String> getParams() {
30              Map<String, String> params = new HashMap<>();
31              params.put("tokenPasien", tokenPasien);
32              params.put("konfirmasi",konfirmasi);
33              params.put("id",id);
34              return params;
35          }
36      };
37      RequestQueue requestQueue = Volley.newRequestQueue(this);
38      requestQueue.add(stringRequest);
39  }
```

5.2Implementasi Node *Subscriber*

Node *subscriber* merupakan web server yang menerima data dari *publisher*. Data yang diterima akan digunakan untuk memperbarui data lokasi dan status ambulans pada basis data. Web server juga melakukan layanan penentuan ambulans berdasarkan kasus pasien dan lokasi pasien. Web server pada penelitian ini menggunakan *framework* Flask yang dituliskan dengan bahasa pemrogaman Python

5.2.1 Implementasi MQTT pada Web Server

Pada Flask, terdapat Flask-MQTT yang merupakan ekstensi Flask untuk protokol MQTT. Tabel 5.15 merupakan potongan baris kode program untuk mengimplementasikan MQTT pada web server. Pada penelitian ini web server adalah *subscriber*, untuk itu perlu dilakukan konfigurasi MQTT client seperti pada baris 11-13. Pada bari 11 mengatur alamat broker, baris 12 mengatur port, dan baris 13 waktu refresh dalam detik.

Baris ke 14 pada Tabel 5.15 merupakan menghubungkan *subscriber* ke *broker*. Pada Flask-MQTT menghubungkan *subscriber* dengan *broker* hanya perlu dengan menginisialisasi ekstensi Flask-MQTT dengan Flask Application. Ini dapat

dilakukan dengan secara langsung melewatkan objek Flask Application pada saat membuat objek.

Subscribe diimplementasikan dengan menggunakan fungsi *subscribe* seperti yang ditunjukan pada baris ke 20 Tabel 5.15 sesuai dengan topik yang dibutuhkan. Pada kode Tabel 5.15 fungsi *subscribe* dipanggil pada fungsi handle_connect, ini dilakukan agar *subscribe* dilakukan setelah *subscriber* sudah terhubung ke *broker*.

Fungsi handle_mqtt_message pada baris 23-29 Tabel 5.15 merupakan fungsi untuk menangani pesan yang di *subscribe*. Pada fungsi tersebut, pesan dalam format JSON akan diubah ke dictionary dan diinisialisasi ke variable-variabel. Selanjutnya akan memanggil fungsi updateDatabase untuk memperbarui data pada basis data berdasarkan data yang diterima dari pesan *subscribe*.

Tabel 5. 15 Implementasi MQTT pada Web Server

Menimplementasikan MQTT pada Web Server

```
1  from flask import Flask
2  from config import Config
3  from flask_mqtt import Mqtt
4  import json
5
6  app = Flask(__name__)
7  app.config.from_object(Config)
8
9  from app import routes
10
11 app.config['MQTT_BROKER_URL'] = 'broker.hivemq.com'
12 app.config['MQTT_BROKER_PORT'] = 1883
13 app.config['MQTT_REFRESH_TIME'] = 1.0
14 mqtt = Mqtt(app)
15
16 from app.subscriber import updateDatabase
17
18 @mqtt.on_connect()
19 def handle_connect(client, userdata, flags, rc):
20     mqtt.subscribe('TestingSkripsiPutri/#')
21
22 @mqtt.on_message()
23 def handle_mqtt_message(client, userdata, message):
24     data = json.loads(message.payload)
25     id = str(data['id'])
26     status = str(data['status'])
27     latitude = str(data['latitude'])
28     longitude = str(data['longitude'])
29     updateDatabase(id, status,latitude,longitude)
```

5.2.2 Implementasi Menerima *Request* dari Aplikasi Pasien

Tabel 5.16 merupakan implementasi fungsi menerima *request* dari aplikasi pasien. Fungsi ini akan memberikan *response* kepada aplikasi pasien berupa ambulans terdekat sesuai dengan kebutuhan pasien. Pada baris ke 12-19, ketika aplikasi pasien mengirimkan *request* POST, maka akan mengambil *value input* dengan nama ‘username’ , ‘nama’, ‘noPonsel’, ‘alamat’, ‘kasus’, ‘latitude’, dan ‘longitude’ untuk di inisialisasi ke dalam variabel.

Pada baris ke 20 Tabel 5.16 memanggil method getAmbulans. Method getAmbulans akan mengembalikan array yang berisi data ambulans yang sesuai dengan kebutuhan pasien berdasarkan lokasi dan kasus. Nilai kembalian dari method ambulans akan diinisialisasi pada variable result. Dari hasil yang didapatkan pada variable result, dilakukan seleksi kondisi apakah variabel result bernilai "{}", jika tidak maka akan menginisialisasi variable dictionary response dengan *key* "username", "success", "message", dan "ambulans" sesuai dengan kode pada Tabel 5.16 berikut. Jika tidak, maka hanya menginisialisasi *key* "success" dan "message".

Tabel 5. 16 Implementasi Menerima *Request* dari Aplikasi Pasien

Fungsi Menerima Request dari Aplikasi Pasien

```
1   @app.route('/getAmbulans', methods=['GET', 'POST'])
2   def ambulans():
3       username =''
4       nama = ''
5       noPonsel = ''
6       alamat = ''
7       kasus = ''
8       latitude =''
9       longitude = ''
10      response=''
11      result =''
12      if request.method == 'POST':
13          username = request.values.get('username')
14          nama = request.values.get('nama')
15          noPonsel = request.values.get('noPonsel')
16          alamat = request.values.get('alamat')
17          kasus  = request.values.get('kasus')
18          latitude = request.values.get('latitude')
19          longitude = request.values.get('longitude')
20          result = getAmbulans(latitude,longitude,kasus)
21
22          response = {}
23          if result != "{}":
24              response['username'] = str(username)
25              response['success'] = "1"
26              response['message'] = "success"
27              response['ambulans'] = result
28          else :
29              response['success'] = "0"
30              response['message'] = "not available"
31          response = json.dumps(response)
32      return response
```

5.2.3 Implementasi Layanan Penentuan Ambulans

Layanan Penentuan Ambulans diimplementasikan pada fungsi getAmbulans yang memiliki parameter lat, lon dan kasus. Lat merupakan titik latitude lokasi pasien, lon merupakan titik longitude lokasi pasien dan kasus merupakan kasus yang dialami pasien. Pada fungsi getAmbulans Tabel 5.17, akan mencari jenis ambulans yang cocok dengan kasus pasien dengan memanggil fungsi getJenisAmbulans. Hasil kembali dari fungsi getJenisAmbulans akan diinisialisasi pada variabel jenisAmbulans. Setelah itu akan mengambil semua data ambulans dari basis data yang memiliki jenis yang sebelumnya telah ditentukan dan

berstatus “STAND BY” dengan memanggil fungsi getFromDatabase. Hasil kembali dari fungsi getFromDatabase akan diinisialisasi pada variabel ambulans. Data ambulans-ambulans yang di dapatkan dari database akan dibandingkan jaraknya dengan jarak pasien untuk menemukan ambulans terdekat. Terakhir, fungsi getAmbulans akan mengembalikan data ambulansterdekat yang telah diubah ke dalam format JSON dengan memanggil fungsi toJSON2.

Tabel 5. 17 Implementasi Fungsi untuk Penentuan Ambulans

Fungsi Menentukan Ambulans

```
1  def getAmbulans(lat,lon,kasus):
2     jenisAmbulans = getJenisAmbulans(kasus)
3     ambulance = getFromDatabase(jenisAmbulans)
4     ambulansterdekat= nearestambulance(lat,lon,ambulance)
5     try:
6        ambulansterdekatJSON                                    =
   toJSON2(ambulansterdekat[0],ambulansterdekat[1],str(ambulansterdekat[2
   ]),str(ambulansterdekat[3]),ambulansterdekat[4],ambulansterdekat[5])
7     except:
8        return "{}"
9     return(ambulansterdekatJSON)
```

Fungsi penentu jenis ambulans pada Tabel 5.18 untuk menentukan jenis ambulans berdasarkan kasus pasien seperti sakit gawat darurat, sakit tidak gawat darurat dan jenazah.

Tabel 5. 18 Implementasi Fungsi Penentu Jenis Ambulans

Fungsi Menentukan Jenis Ambulans

```
1  def getJenisAmbulans(kasus):
2     if kasus == 'Sakit Gawat Darurat':
3        return 'gawat darurat'
4     elif kasus == 'Sakit tidak Gawat Darurat':
5        return 'transportasi'
6     elif kasus == 'Jenazah':
7        return 'jenazah'
8     else :
9        return ""
```

Fungsi getFromDatabase pada Tabel 5.19 untuk mendapatkan data ambulans dari basis data berdasarkan jenis ambulans.

Tabel 5. 19 Implementasi Fungsi untuk Mendapatkan Ambulans Berdasarkan Jenis

Fungsi Mendapatkan Ambulans Berdasarkan Jenis

```
1  def getFromDatabase(jenis):
2     mydb =connect()
3     mycursor = mydb.cursor()
4
5     sql  ="SELECT  *  FROM  ambulance  WHERE  status  =  'STAND  BY'  AND
   jenis='"+jenis+"'"
6
7     mycursor.execute(sql)
8     myresult = mycursor.fetchall()
9     return myresult
```

Fungsi nearestAmbulance pada Tabel 5.20 merupakan fungsi untuk mencari ambulans dengan jarak terdekat dari pasien. Pencarian ambulans terdekat dilakukan dengan melakukan perulangan untuk membandingkan jarak ambulans –ambulans tersebut dengan pasien.

Tabel 5. 20 Implementasi Fungsi untuk Mencari Ambulans Terdekat

Fungsi Mencari Ambulans Terdekat	
1	def nearestambulance(lat,lon,ambulance):
2	nearestambulance=''
3	shortestdistance=float("inf")
4	for x in ambulance:
5	distance = haversine(x[2],x[3],float(lat),float(lon))
6	if distance<shortestdistance:
7	nearestambulance = x
8	shortestdistance = distance
9	result =(nearestambulance[0],nearestambulance[1],nearestambulance[2],nearestambulance[3],nearestambulance[4],str(shortestdistance))
10	return(result)

Fungsi haversine pada Tabel 5.21 adalah fungsi untuk menghitung jarak dua titik koordinat ke dalam kilometer menggunakan formula haversine.

Tabel 5. 21 Implementasi Fungsi untuk Menghitung Jarak Dua Titik Koordinat

Fungsi Menghitung Jarak dua Titik Koordinat	
1	def haversine(lat1,lon1, lat2, lon2):
2	lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
3	dlon = lon2 - lon1
4	dlat = lat2 - lat1
5	a = sin(dlat / 2) ** 2 + cos(lat1) * cos(lat2) * sin(dlon / 2) ** 2
6	d= 2 * 6371 * asin(sqrt(a))
7	return d

Fungsi toJSON2 Tabel 5.22 adalah fungsi untuk mengolah data yang di inputkan menjadi data dalam format JSON

Tabel 5. 22 Implementasi untuk Mengolah Data ke Format JSON

Fungsi Mengolah Data ke Format JSON	
1	def toJSON2(id, hos,lat,lon,jenis,jarak):
2	return {"id":id,"ket":hos,"lat":lat,"lon":lon,"jenis":jenis,"jarak":jarak}

5.3 Implementasi Node Aplikasi Perangkat Bergerak Pasien

Node aplikasi perangkat bergerak pasien adalah aplikasi android yang digunakan oleh *user* untuk mengirimkan permintaan kepada webserver. Permintaan yang dikirimkan merupakan permintaan untuk mengetahui ambulans yang sesuai dengan user berdasarkan kasus dan lokasi.

5.3.1 Implementasi *Subscribe*

Tabel 5.23 merupakan ptongan baris kode program fungsi untuk melakukan subscribe. Pada baris ke 3 merupakan topik yang akan di subscribe. Topik yang di subscribe ditambahkan ID ambulans untuk melakukan subscribe pada ambulans yang dibutuhkan saja. Kemudian pada baris ke 4 akan memanggil method subscribe dengan memasukan topik. Jika subscribe berhasil maka akan memanggil method onSuccess pada baris ke 6 dan jika gagal akan memanggil method onFailure pada baris ke 11.

Tabel 5. 23 Implementasi Fungsi Subscribe

Fungsi Mengolah Data ke Format JSON

```
1   private void subscribeToTopic() {
2       try {
3           topik = "TestingSkripsiPutri/"+ID_Ambulans;
4           client.subscribe(topik, 0, null, new IMqttActionListener() {
5               @Override
6               public void onSuccess(IMqttToken asyncActionToken) {
7                   Log.w("Mqtt","Subscribed!");
8               }
9
10              @Override
11              public   void   onFailure(IMqttToken   asyncActionToken,
    Throwable exception) {
12                  Log.w("Mqtt", "Subscribed fail!");
13              }
14          });
15
16      } catch (MqttException ex) {
17          System.err.println("Exceptionst subscribing");
18          ex.printStackTrace();
19      }
20  }
```

5.3.2 Implementasi *Event Handler*

Tabel 5.24 merupakan potongan baris kode program yang dipanggil ketika menekan tombol “Pilih”. Tombol “Pilih” merupakan tombol yang di tekan ketika *user* membutuhkan ambulans dan telah memasukan data diri *user* beserta kasus. Pada baris 2-5 mengambil data yang di masukan oleh *user*. Kemudian pada baris ke 7 dilakukan seleksi kondisi apakah terdapat *form* masukan yang kosong. Jika iya, maka akan memberikan pesan *error* untuk melengkapi *form* masukan. Jika tidak, aplikasi akan mencari lokasi saat ini dengan menggunakan FusedLocationProviderClient adalah layanan Google Location Service API. Pada baris ke 8 dilakukan inisialisasi FusedLocationProviderClient, kemudian untuk mendapatkan lokasi saat ini, digunakan getLastLocation() API seperti pada baris ke 9. Yang ditambahkan success callback listener yang akan terpanggil ketika lokasi didapatkan. Ketika lokasi didapatkan maka akan melakukan seleksi kondisi apakah lokasi yang di dapatkan kosong. Jika tidak maka titik koordinat berhasil didapatkan. Saat titik koordinat berhasil didapatkan, method getAmbulans dipanggil untuk mengirimkan request ke web server.

Tabel 5. 24 Implementasi Fungsi *Event Handler*

Fungsi Event Handler

```
1  public void btn_ambulans_onClick(View view){
2      final String mNama = editText_nama.getText().toString();
3      final String mNoPonsel = editText_noPonsel.getText().toString();
4      final String mAlamat = editText_alamat.getText().toString();
5      final String mKasus = spinner_kasus.getSelectedItem().toString();
6
7      if(!mNama.isEmpty()||!mAlamat.isEmpty()||!mNoPonsel.isEmpty()){
8          FusedLocationProviderClient mFusedLocation =
   LocationServices.getFusedLocationProviderClient(MainActivity.this);
9          mFusedLocation.getLastLocation().addOnSuccessListener( new
   OnSuccessListener<Location>() {
10             @Override
11             public void onSuccess(Location location) {
12                 if (location != null) {
13                     myLatitude =
   String.valueOf(location.getLatitude());
14                     myLongitude =
   String.valueOf(location.getLongitude());
15                     getAmbulans(mUsername ,mNama,mNoPonsel
   ,mKasus,mAlamat , myLatitude,myLongitude);
16                 }
17             }
18         });
19
20     }
21     else{
22         if(mNama.isEmpty()){
23             editText_nama.setError("Masukkan nama");
24         }
25         if(mAlamat.isEmpty()){
26             editText_alamat.setError("Masukkan alamat");
27         }
28         if(mNoPonsel.isEmpty()){
29             editText_noPonsel.setError("Masukan nomor ponsel");
30         }
31
32     }
33 }
```

5.3.3 Implementasi Mengirimkan *Request* ke Web Server

Request dikirimkan dengan menggunakan *library* volley. Volley adalah *library* yang mengelola permintaan dan respon tanpa harus menulis banyak kode. Pada baris 3-57 Tabel 5.25 adalah membuat POST *request* dengan parameter "username", "nama", "noPonsel", "kasus", "alamat", "latitude","longitude" dan *response* String. Jika tidak terdapat *error* maka akan masuk ke method onResponse, jika error akan ke method onErrorResponse.

Pada method onResponse di Tabel 5.25 baris ke 9-18, akan melakukan konversi String response ke JSON object. Dari JSON Object tersebut didapatkan String "success" yang di inisialisasi pada variabel success dan JSON object "ambulans" yang di inisialisasi pada variable jsonObjectData. Untuk mengecek apakah web server sukses memberikan *response* ambulans yang dibutuhkan pasien, dilakukan seleksi kondisi apakah nilai success sama dengan "1". Jika tidak akan menampilkan pada halaman utama bahwa ambulans tidak tersedia, jika iya

maka akan mendapatkan String "ket" ,"id", "lat", "lon", dan "jarak" dari jsonObjectData yang akan ditampilkan pada popup dengan memanggil fungsi ShowPopup.

Setelah selesai membuat request, maka seperti yang ditunjukan pada Tabel 5.25 baris 59-60, menginisialisasi RequestQueue dan menambahkan *request* yang dibuat pada RequestQeueu.

Tabel 5. 25 Implementasi Fungsi untuk Mengirimkan *Request* ke Web Server

```
Fungsi Mengirimkan Request ke Web Server
1   public void getAmbulans(final String username,final String nama,final
    String noPonsel,final String kasus,final String alamat, final String
    latitude, final String longitude){
2       loading.setVisibility(View.VISIBLE);
3
4       StringRequest           stringRequest           =           new
    StringRequest(Request.Method.POST,          URL_AMBULANS,          new
    Response.Listener<String>() {
5           @Override
6           public void onResponse(String response) {
7               try{
8                   Log.d("Response",response);
9                   JSONObject jsonObject = new JSONObject(response);
10                  String success = jsonObject.getString("success");
11                  JSONObject              jsonObjectData              =
    jsonObject.getJSONObject("ambulans");
12
13                  if(success.equals("1")){
14                      ketPopup = jsonObjectData.getString("ket");
15                      IDAmbulans = jsonObjectData.getString("id");
16                      latitudePopup = jsonObjectData.getString("lat");
17                      longitudePopup = jsonObjectData.getString("lon");
18                      jarakPopup = jsonObjectData.getString("jarak");
19
20                      . . .
21
22                      ShowPopup(ketPopup,
    latitudePopup,longitudePopup,jarakPopup);
23
24                      . . .
25                  }
26                  else{
27                      Toast.makeText(MainActivity.this,       "Ambulans
    sedang tidak tersedia",Toast.LENGTH_SHORT).show();
28                  }
29              }
30              catch (JSONException e){
31                  e.printStackTrace();
32                  Toast.makeText(MainActivity.this,  "JSON  Exception  :
    "+e,Toast.LENGTH_SHORT).show();
33              }
34          }
35      }, new Response.ErrorListener() {
36          @Override
37          public void onErrorResponse(VolleyError error) {
38              Toast.makeText(MainActivity.this,      "VolleyError      :
    "+error,Toast.LENGTH_SHORT).show();
39
40          }
41      })
42      {
43          // POST request dengan data berikut
```

```
44              @Override
45              protected Map<String, String> getParams() {
46                  Map<String, String> params = new HashMap<>();
47                  params.put("username", username);
48                  params.put("nama",nama);
49                  params.put("noPonsel",noPonsel);
50                  params.put("kasus",kasus);
51                  params.put("alamat",alamat);
52                  params.put("latitude",latitude);
53                  params.put("longitude",longitude);
54
55                  return params;
56              }
57          };
58          Log.d("Request",String.valueOf(stringRequest));
59          RequestQueue requestQueue = Volley.newRequestQueue(this);
60          requestQueue.add(stringRequest);
61      }
```

BAB 6 PENGUJIAN SISTEM

Pengujian sistem dilakukan setelah melakukan implementasi. Pengujian sistem dilakukan untuk menguji sistem yang dibuat, apakah sistem memenuhi kebetuhan fungsional. Pengujian sistem juga dilakukan untuk mengetahui kinerja dari sistem. Pengujian sistem meliputi pengujian fungsional dan pengujian kinerja. Pengujian fungsional dilakukan didasari oleh kebutuhan fungsional. Pengujian kinerja dilakukan dengan menghitung waktu pengiriman data yang dibutuhkan dari *publisher* ke *subscriber*.

6.1 Pengujian Fungsional

Pengujian fungsional adalah pengujian yang dilakukan untuk memastikan sistem memenuhi kebutuhan fungsional sistem.

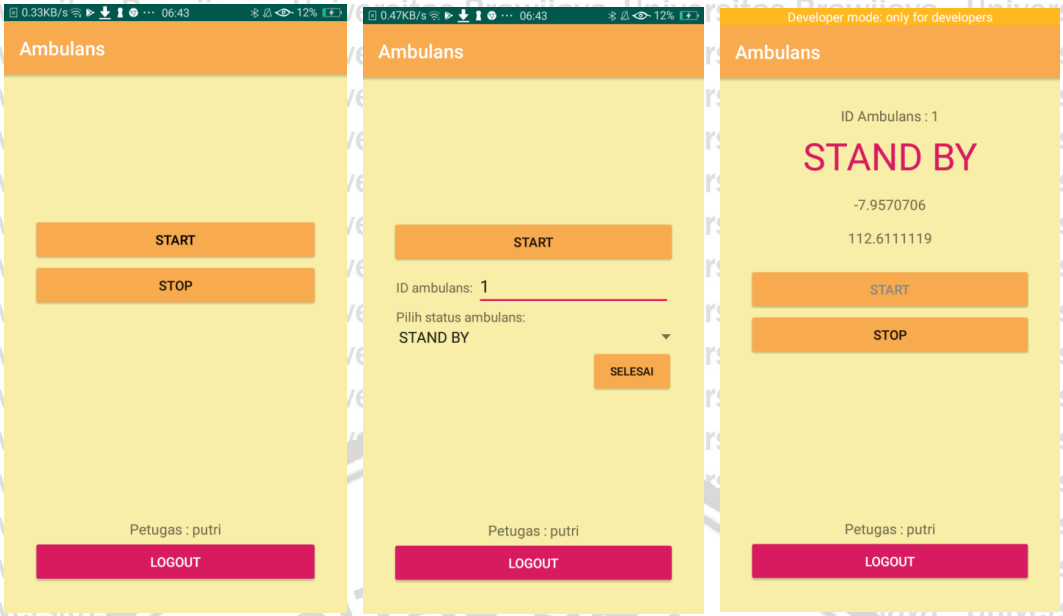
6.1.1 Pengujian Kode KF-01-01

Pengujian kode KF-01-01 dilakukan dengan senario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.1.

Tabel 6. 1 Tabel Pengujian Fungsional KF-01-01

Kode Fungsi	KF-01-01
Kebutuhan Fungsional	Node *publisher* harus dapat menentukan lokasi dari perangkat node menggunakan GPS
Skenarion Pengujian	1. Menyalakan *location service* pada perangkat node *publisher* 2. Node *publisher* mengimplementasikan Google Play Service Location API untuk mendapatkan lokasi perangkat dalam bentuk koordinat 3. Menampilkan lokasi yang didapatkan berupa koordinat pada halaman utama
Hasil yang Diharapkan	Node *publisher* mendapatkan lokasi koordinat dan menampilkan lokasi yang didapatkan pada halaman utama
Hasil Pengujian	Node *publisher* berhasil mendapatkan lokasi koordinat dan menampilkan lokasi yang didapatkan pada halaman utama

Hasil pengujian fungsional kode KF-01-01 adalah node *publisher* berhasil mendapatkan lokasi perangkat saat ini dan menampilkannya pada halaman utama aplikasi. Node publisher dijalankan dalam keadaan berpindah-pindah tempat untuk melihat apakah titik koordinat berubah seiring dengan perpindahan lokasi node publisher. Gambar 6.1 menampilkan tampilan aplikasi *publisher* ketika mendapatkan lokasi titik koordinat perangkat. Titik koordinat yang didapatkan adalah titik koordinat garis bujur (*longitude*) dan titik koordinat garis lintang (*latitude*).



Gambar 6. 1 Tampilan Aplikasi Perangkat Bergerak Ambulans (*Publisher*)

6.1.2 Pengujian Kode KF-01-02

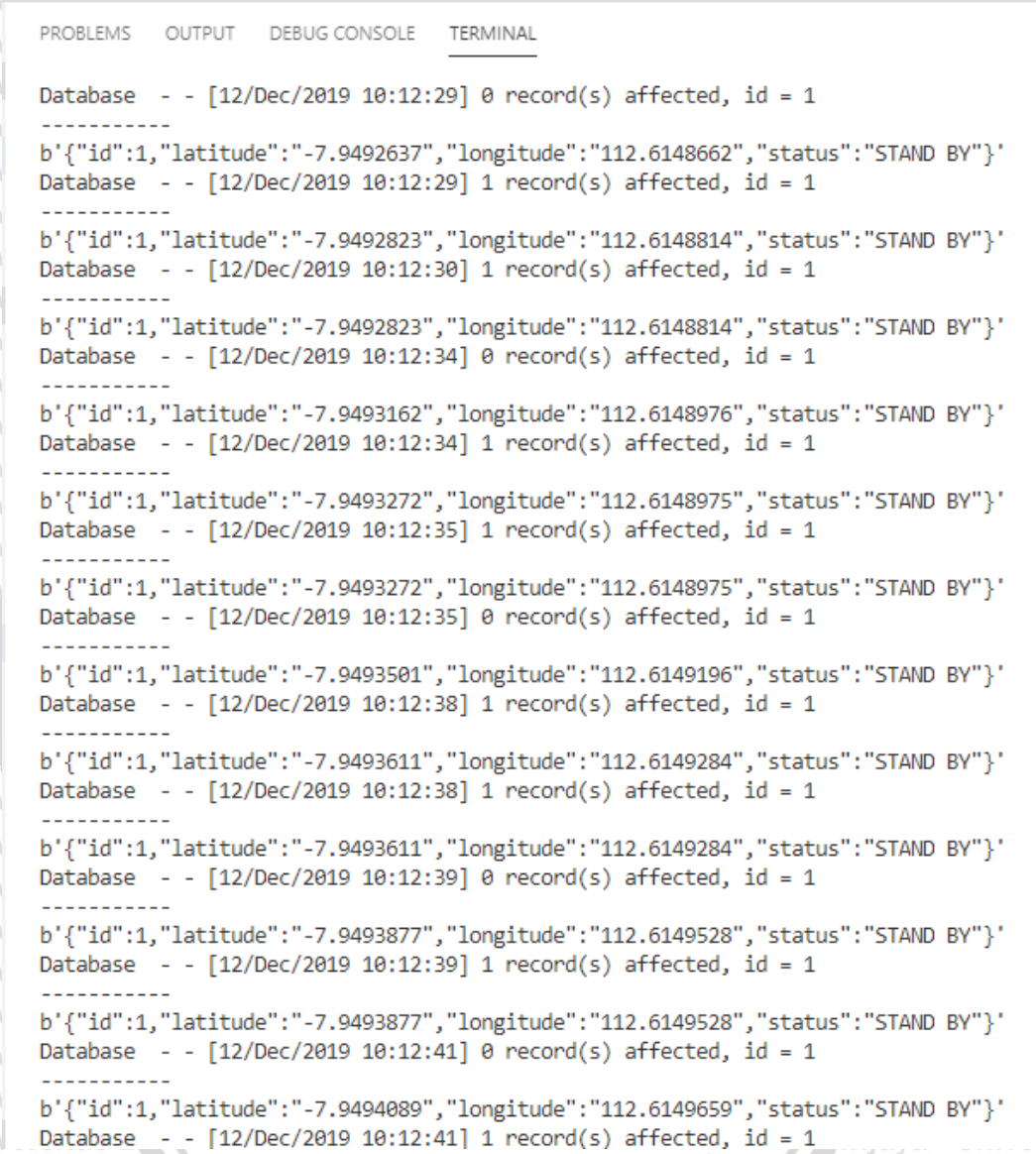
Pengujian kode KF-01-02 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.2.

Tabel 6. 2 Tabel Pengujian Fungsional KF-01-02

Kode Fungsi	KF-01-02
Kebutuhan Fungsional	Node *publisher* harus dapat melakukan *publish* data secara terus menerus ke broker dengan format JSON
Skenarion Pengujian	1. Node *publisher* menghubungkan *publisher* dengan *broker* 2. Node *publisher* melakukan *publish* 3. Menampilkan data yang diterima pada *subscriber*
Hasil yang Diharapkan	*Publisher* dapat melakukan *publish* dan *subscriber* dapat menampilkan data yang di *publish* oleh *publisher* dengan format JSON
Hasil Pengujian	*Publisher* berhasil melakukan *publish* dan *subscriber* berhasil menampilkan data yang di *publish* oleh *publisher* dengan format JSON

Hasil pengujian kode KF-01-02 adalah *publisher* berhasil melakukan *publish* dan *subscriber* berhasil menampilkan data yang di *publish* oleh *publisher* dengan format JSON. Node publisher mengirimkan data lokasi dalam keadaan bergerak sehingga data lokasi yang diterima oleh subscriber berubah-ubah sesuai dengan lokasi node publisher. Gambar 6.2 adalah terminal node subscriber yang

menerima data. Pada terminal node *subscriber* akan mencetak data yang diterima dari *publisher*. Pada gambar tersebut terlihat bahwa data yang dikirimkan oleh *publisher* berhasil diterima oleh *subscriber*.



Gambar 6. 2 Terminal pada Node *Subscriber* Menampilkan Data yang Didapatkan

6.1.3 Pengujian Kode KF-01-03

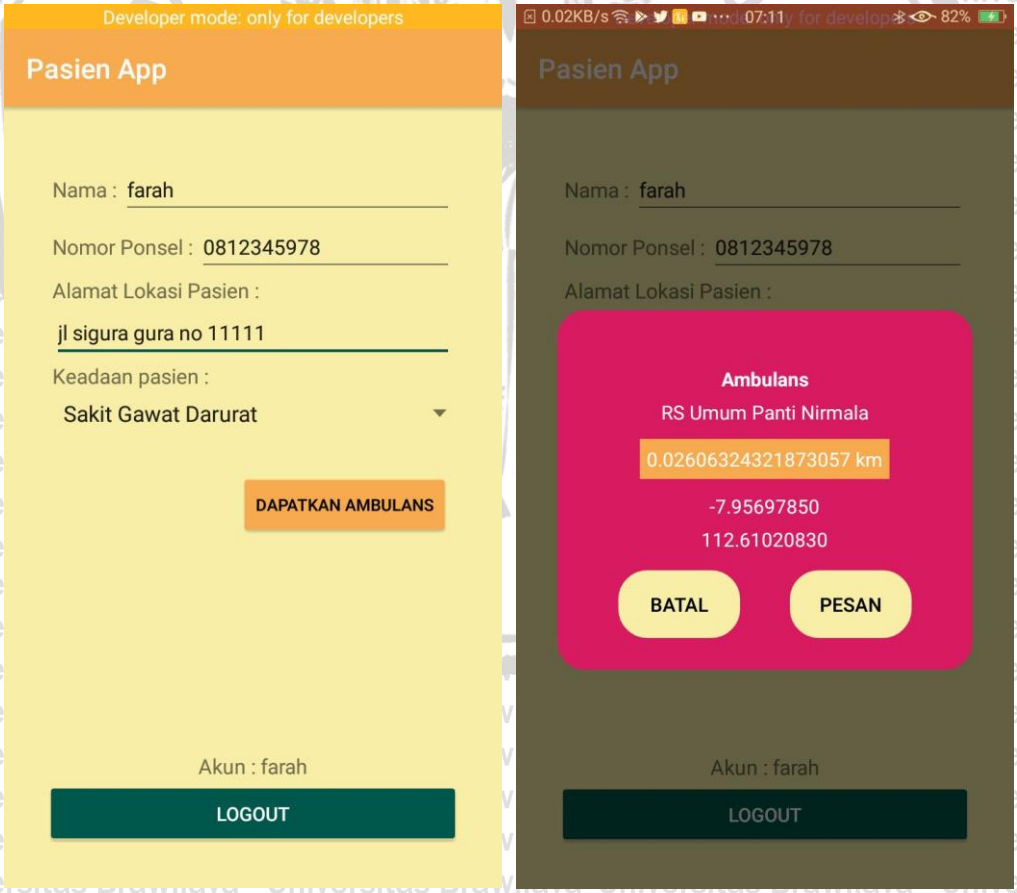
Pengujian kode KF-01-03 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.3.

Tabel 6. 3 Tabel Pengujian Fungsional KF-01-03

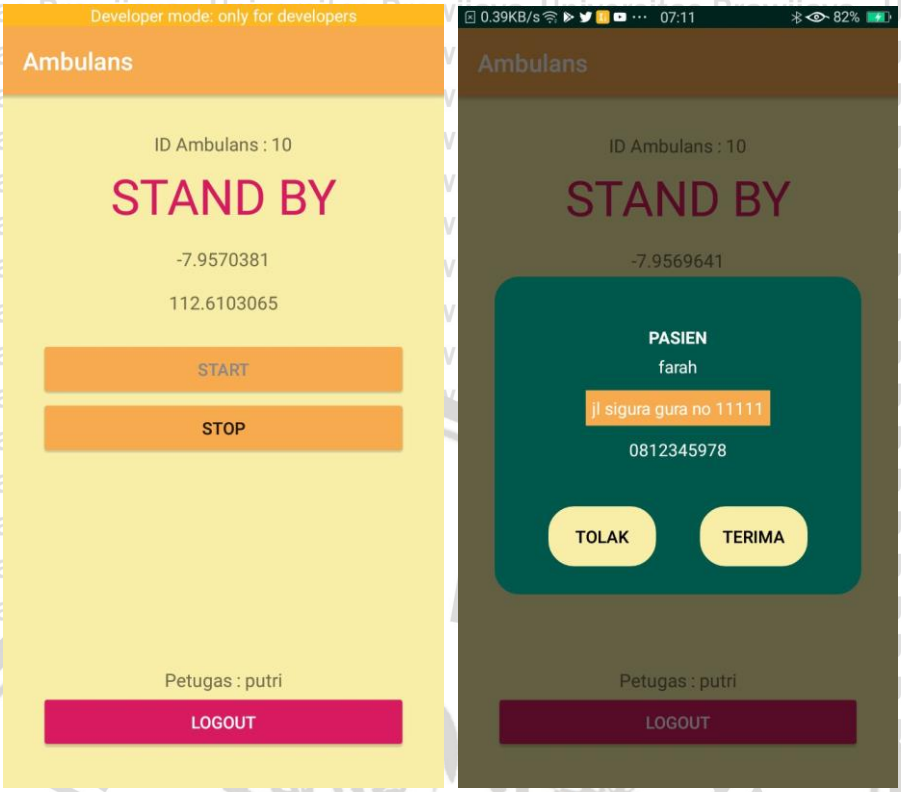
Kode Fungsi	KF-01-03
Kebutuhan Fungsional	Node publisher harus dapat menerima informasi pasien yang membutuhkan ambulans

Skenarion Pengujian	1. Node *subscriber* sebagai web server mengirimkan informasi pasein ke ambulans 2. Node *publisher* menampilkan data yang dikirimkan node *subscriber*
Hasil yang Diharapkan	Node *publisher* mendapatkan data informasi pasien yang dikirimkan oleh web server node subscriber dan menampilkan data tersebut
Hasil Pengujian	Node *publisher berhasil* mendapatkan data informasi pasien yang dikirimkan oleh web server node *subscriber* dan menampilkan data tersebut

Hasil pengujian kode KF-01-03 Node *publisher berhasil* mendapatkan data informasi pasien yang dikirimkan oleh web server node *subscriber* dan menampilkan data tersebut. Pada Gambar 6.3 tampilan aplikasi pasien yang melakukan permintaan ambulans. Kemudian pada Gambar 6.4 merupakan aplikasi node publisher ambulans yang berhasil menerima data informasi yang dikirimkan oleh web server.



Gambar 6. 3 Aplikasi Pasien



Gambar 6. 4 Aplikasi Node Publisher Menampilkan Informasi Pasien

6.1.4 Pengujian Kode KF-01-04

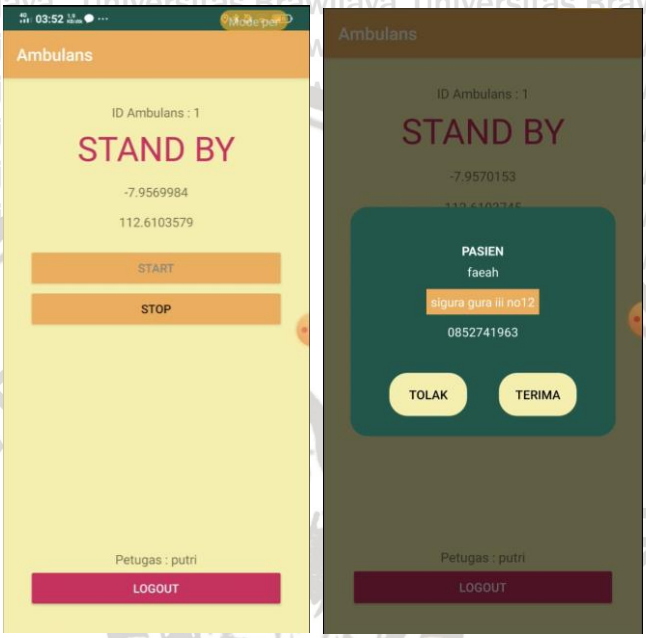
Pengujian kode KF-01-04 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.4.

Tabel 6. 4 Tabel Pengujian Fungsional KF-01-04

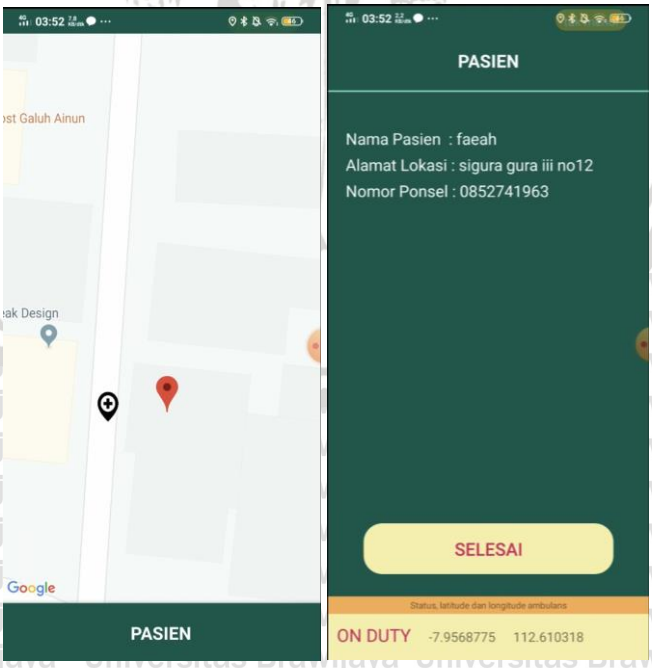
Kode Fungsi	KF-01-04
Kebutuhan Fungsional	Node *publisher* harus dapat menolak atau menerima melakukan penjemputan pasien
Skenarion Pengujian	1. Node *subscriber* sebagai web server mengirimkan informasi pasein ke ambulans 2. Node publisher menampilkan informasi pasien 3. Node publisher menampilkan tombol untuk menolak atau menerima penjemputan pasien 4. Node *publisher* mengirimkan konfirmasi menolak atau menerima penjemputan pasien ke node *subscriber* yang merupakan web server 5. Node *subscriber* sebagai web server menampilkan konfirmasi dari node *publisher*
Hasil yang Diharapkan	Node *publisher* dapat menerima atau menolak melakukan penjemputan pasien
Hasil Pengujian	Node *publisher* berhasil menerima atau menolak melakukan

	penjemputan pasien

Hasil pengujian kode KF-01-04 Node *publisher* dapat menerima atau menolak melakukan penjemputan pasien. Pada Gambar 6.5 merupakan aplikasi node publisher ambulans yang berhasil menerima data informasi yang dikirimkan oleh web server. Selanjutnya pada Gambar 6.6 adalah tampilan aplikasi publisher ambulans ketika menerima permintaan penjemputan pasien, aplikasi akan membuka halaman baru yang berisi *maps* dan data pasien.

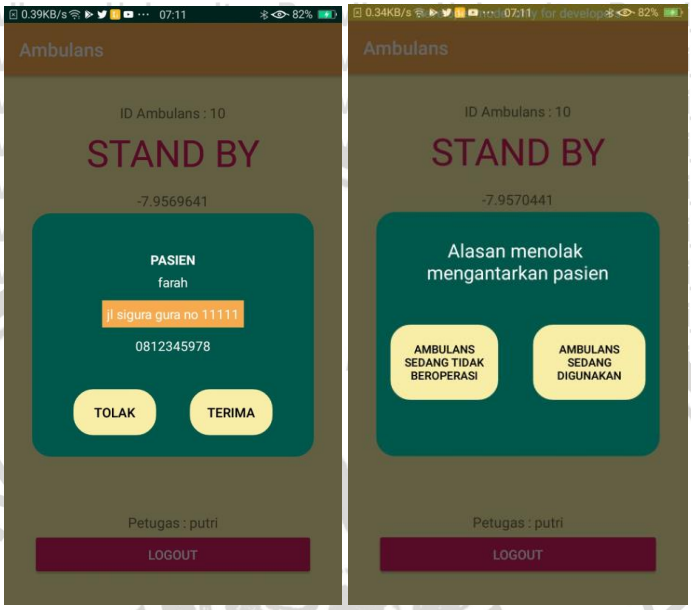


Gambar 6. 5 Tampilan Aplikasi Publisher untuk Menolak atau Menerima

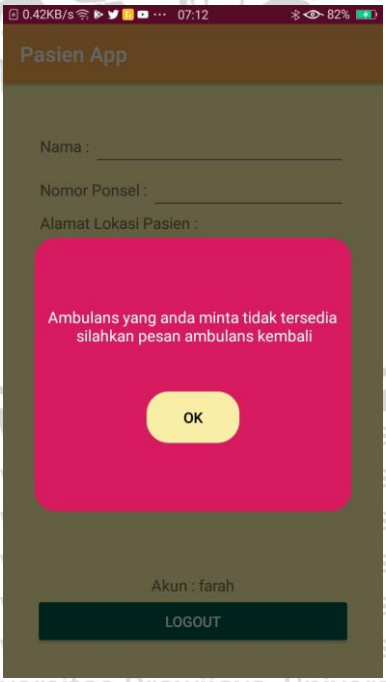


Gambar 6. 6 Aplikasi Publisher Menerima Penjemputan Pasien

Pada Gambar 6.7 merupakan aplikasi node publisher ambulans ketika memilih tombol “tolak”. Ketika memilih tombol tolak maka akan menampilkan pilihan alasan menolak mengantarkan pasien. Gambar 6.8 merupakan aplikasi pasien ketika aplikasi publisher ambulans menolak melakukan penjemputan, maka pada aplikasi pasien akan muncul keterangan bahwa ambulans sedang tidak tersedia.



Gambar 6. 7 Aplikasi Publisher Menolak Penjemputan Pasien



Gambar 6. 8 Aplikasi Pasien Menampilkan Keterangan

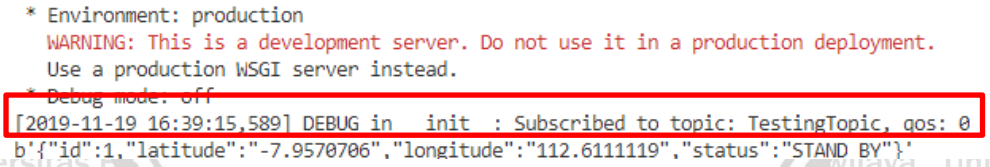
6.1.5 Pengujian Kode KF-02-01

Pengujian kode KF-02-01 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.5.

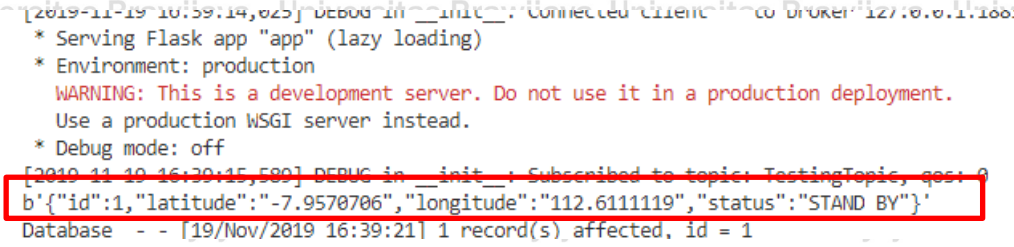
Tabel 6. 5 Tabel Pengujian Fungsional KF-02-01

Kode Fungsi	KF-02-01
Kebutuhan Fungsional	Node *subscriber* harus dapat melakukan *subscribe* ke *broker*
Skenarion Pengujian	1. Node *subscriber* menghubungkan *subscriber* dengan *broker* 2. Menampilkan bahwa berhasil melakukan *subscribe* 3. Node *subscriber* menerima data sesuai dengan topik
Hasil yang Diharapkan	Node *subscriber* melakukan *subscribe* topik sehingga mendapatkan data yang dikirim oleh *publisher* serta menampilkan data tersebut
Hasil Pengujian	Node *subscriber* berhasil melakukan *subscribe* topik sehingga mendapatkan data yang dikirim oleh *publisher* serta menampilkan data tersebut

Hasil pengujian kode KF-02-01 adalah Node *subscriber* berhasil melakukan *subscribe* topik sehingga mendapatkan data yang dikirim oleh *publisher* serta menampilkan data tersebut. Pada Gambar 6.9 menampilkan terminal node *subscriber* yang berhasil melakukan *subscribe* pada topik "TestingTopic", dan pada Gambar 6.10 merupakan gambar terminal node *subscriber* yang berhasil menerima data dari *publisher*. Node *subscriber* yang menerima data dari *publisher* berarti node *subscriber* berhasil melakukan *subscribe* topik.



Gambar 6. 9 Terminal pada Node *Subscriber* Menampilkan *Subsciber* Berhasil *Subscribe*



Gambar 6. 10 Terminal pada Node *Subscriber* Menampilkan data *Subscribe*

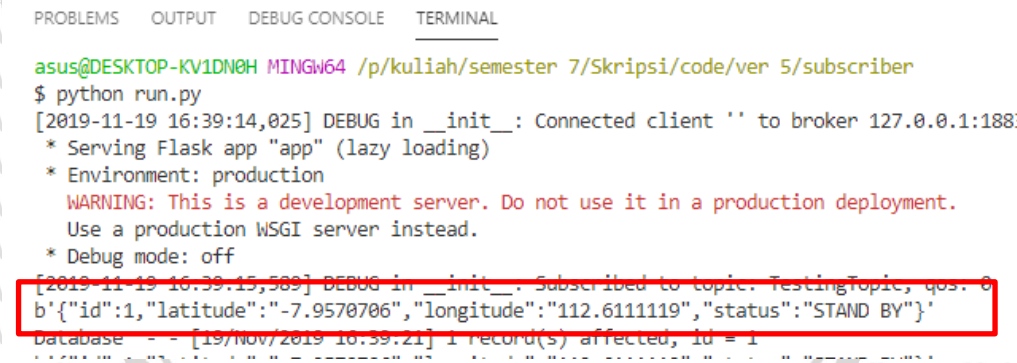
6.1.6 Pengujian Kode KF-02-02

Pengujian kode KF-02-02 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.6.

Tabel 6. 6 Tabel Pengujian Fungsional KF-02-02

Kode Fungsi	KF-02-02
Kebutuhan Fungsional	Node *subscriber* harus dapat menerima data dari *publisher* yang diteruskan oleh broker
Skenarion Pengujian	1. Node *subscriber* menghubungkan *subscriber* dengan *broker* 2. Menampilkan data yang diterima ketika menerima data
Hasil yang Diharapkan	Node *subscriber* mendapatkan data yang dikirim oleh *publisher* dan menampilkan data tersebut
Hasil Pengujian	Node *subscriber* berhasil mendapatkan data yang dikirim oleh *publisher* dan menampilkan data tersebut

Hasil pengujian kode KF-02-02 adalah Node *subscriber* berhasil mendapatkan data yang dikirim oleh *publisher* dan menampilkan data tersebut. Pada Gambar 6.11 menampilkan terminal node *subscriber* yang berhasil menerima data dari *publisher*.



Gambar 6. 11 Terminal pada Node *Subscriber* Menampilkan Data yang Didapatkan

6.1.7 Pengujian Kode KF-02-03

Pengujian kode KF-02-03 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.7.

Tabel 6. 7 Tabel Pengujian Fungsional KF-02-03

Kode Fungsi	KF-02-03
Kebutuhan Fungsional	Node *subscriber* harus dapat melakukan pembaruan data yang diterima dari *publisher* pada basis data

Skenarion Pengujian	1. Menjalankan node *subscriber* dan melakukan *subscribe* 2. Menjalankan *broker* 3. Menjalankan program *publisher* untuk mengirimkan data 4. Node *subscriber* menerima data yang diteruskan oleh *broker* dari *publisher* 5. Node *subscriber* melakukan pembaruan pada basisdata berdasarkan data yang diterima 6. Menampilkan data yang diperbarui
Hasil yang Diharapkan	Data pada basis data diperbarui
Hasil Pengujian	Data pada basis data berhasil diperbarui

Hasil pengujian kode KF-02-03 basis data berhasil diperbari. Pada Gambar 6.12 menampilkan terminal node *subscriber* yang menampilkan data yang diterima dan pemberitahuan bahwa 1 baris pada basis data berhasil diperbarui. Gambar 6.13 adalah basis data sebelum diperbarui yang memiliki nilai latitude "-7.95550260", longitude "112.60193770" dan status "STAND BY". Pada Gambar 6.14 adalah basis data setelah diperbarui. Pada gambar tersebut terlihat bahwa nilai latitude, longitude dan status sama dengan data yang diterima dari publisher yaitu nilai latitude "-7.9570706", longitude "112.6111119" dan status "STAND BY".

```
asus@DESKTOP-KV1DN0H MINGW64 /p/kuliah/semester 7/Skripsi/code/ver 5/subscriber
$ python run.py
[2019-11-19 16:39:14,025] DEBUG in __init__: Connected client '' to broker 127.0.0.1:1883
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
[2019-11-19 16:39:15,589] DEBUG in __init__: Subscribed to topic: TestingTopic, qos: 0
b'{"id":1,"latitude":"-7.9570706","longitude":"112.6111119","status":"STAND BY"}'
Database  - - [19/Nov/2019 16:39:21] 1 record(s) affected, id = 1
```

Gambar 6. 12 Terminal Node *Susbcriber* Menampilkan Berhasil Memperbarui Data

ID ▲ 1	hospital	latitude	longitude	status	jenis
1	RS Universitas Brawijaya	-7.95550260	112.60193770	STAND BY	gawat darurat
2	RS Umum Daerah Kota Malang	-7.99127950	112.63464870	ON DUTY	transportasi
3	RSU Persada Hospital	-7.96821580	112.60393800	STAND BY	jenazah
4	RSU Hermina Tangkubanprahu	-7.97806840	112.62233500	OFF DUTY	gawat darurat
5	Rumah Sakit Islam UNISMA Malang	-7.94026620	112.60662950	OFF DUTY	gawat darurat
6	RSU Islam Aisyiyah Malang	-7.98353070	112.62257480	STAND BY	jenazah
7	RS Umum Universitas Muhammadiyah Malang	-7.93543200	112.70675080	STAND BY	transportasi
8	RS Umum Lavalette Malang	-7.96584900	112.63797200	STAND BY	gawat darurat

Gambar 6. 13 Basis Data Sebelum Data Diperbarui

ID ▲ 1	hospital	latitude	longitude	status	jenis
1	RS Universitas Brawijaya	-7.95707060	112.61111190	STAND BY	gawat darurat
2	RS Umum Daerah Kota Malang	-7.99127950	112.63464870	ON DUTY	transportasi
3	RSU Persada Hospital	-7.96821580	112.60393800	STAND BY	jenazah
4	RSU Hermina Tangkubanprahu	-7.97806840	112.62233500	OFF DUTY	gawat darurat
5	Rumah Sakit Islam UNISMA Malang	-7.94026620	112.60662950	OFF DUTY	gawat darurat
6	RSU Islam Aisyiyah Malang	-7.98353070	112.62257480	STAND BY	jenazah
7	RS Umum Universitas Muhammadiyah Malang	-7.93543200	112.70675080	STAND BY	transportasi
8	RS Umum Lavalette Malang	-7.96584900	112.63797200	STAND BY	gawat darurat
9	RS Umum Panti Waluya Sawahan Malang	-7.98656170	112.62277880	STAND BY	gawat darurat
10	RS Umum Panti Nirmala	-7.91561710	112.60439780	STAND BY	transportasi
11	RS Umum Daerah Dr. Saiful Anwar	-7.97256200	112.62935690	STAND BY	jenazah

Gambar 6. 14 Basis Data Setelah Data Diperbarui

6.1.8 Pengujian Kode KF-02-04

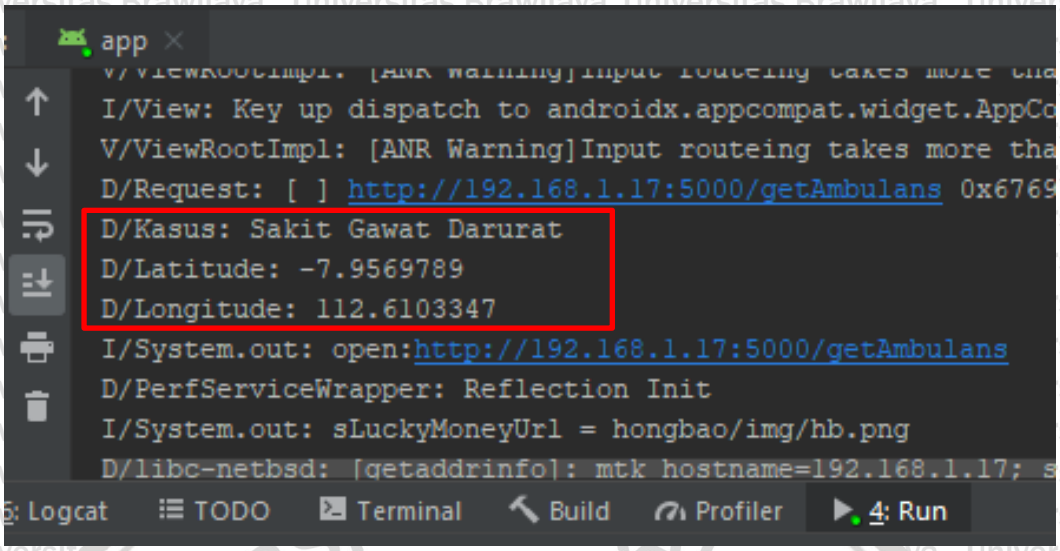
Pengujian kode KF-02-04 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.8.

Tabel 6. 8 Tabel Pengujian Fungsional KF-02-04

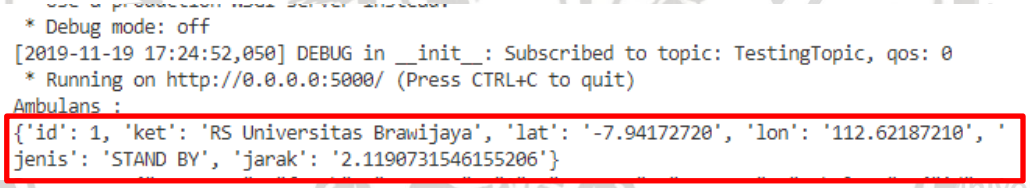
Kode Fungsi	KF-02-04
Kebutuhan Fungsional	Node *subscriber* harus dapat melakukan penentuan ambulans yang dibutuhkan oleh pasien berdasarkan status ambulans, kasus pasien, dan lokasi ambulans
Skenarion Pengujian	1. Node *subscriber* menerima input lokasi pasien, dan kasus pasien 2. Node *subscriber* menjalankan fungsi untuk menentukan jenis ambulans berdasarkan status ambulans dan jarak ambulans 3. Menampilkan hasil penentuan ambulans
Hasil yang Diharapkan	Node *subscriber* menentukan ambulans untuk pasien berdasarkan lokasi koordinat dan status pasien dan menampilkan data ambulans tersebut
Hasil Pengujian	Node *subscriber* berhasil menentukan ambulans untuk pasien berdasarkan lokasi koordinat dan status pasien dan menampilkan data ambulans tersebut

Hasil pengujian kode KF-02-04 adalah Node *subscriber* berhasil menentukan ambulans untuk pasien berdasarkan lokasi koordinat dan status pasien dan menampilkan data ambulans tersebut. Pada Gambar 6.15 adalah pesan log aplikasi pasien yang menampilkan latitude, longitude dan kasus pasien. Gambar

6.16 adalah gambar terminal node subscriber setelah menentukan ambulans berdasarkan data pada Gambar 6.15.



Gambar 6. 15 Pesan Log Aplikasi Pasien Menampilkan Latitude, Longitude dan Kasus yang Dikirimkan



Gambar 6. 16 Terminal Node *Subscriber* Menampilkan Hasil Penentuan Ambulans

Tabel 6.7 adalah data ambulans yang ada pada basis data dan perhitungan jarak ambulans-ambulans tersebut dengan pasien berdasarkan koordinat yang dikirimkan pada Gambar 6.15, dimana kasus pasien adalah sakit gawat darurat. Pasien sakit gawat darurat akan mendapatkan ambulans dengan jenis gawat darurat. Ambulans yang dapat digunakan oleh pasien hanya ambulans dengan status STAND BY yang berarti ambulans siap untuk dgunakan. Oleh karena itu ambulans yang memenuhi kasus pasien dan tersedia adalah ambulans dengan id 1, 8, 9, 14 dan 15. Dari 5 ambulans tersebut, ambulans dengan jarak terdekat adalah ambulans dengan id 1.

Tabel 6. 9 Tabel Data Ambulans dan Jarak Ambulans dengan Pasien

id	ket	latitude	longitude	status	jenis	Jarak
1	RS Universitas Brawijaya	-7.94172720	112.62187210	STAND BY	gawat darurat	2.1190731546155206

2	RS Umum Daerah Kota Malang	-7.99127950	112.63464870	ON DUTY	transportasi	4.660015704757043
3	RSU Persada Hospital	-7.96821580	112.60393800	STAND BY	jenazah	1.434373590514046
4	RSU Hermina Tangkuban prahu	-7.97806840	112.62233500	OFF DUTY	gawat darurat	2.691761084948309
5	Rumah Sakit Islam UNISMA Malang	-7.94026620	112.60662950	OFF DUTY	gawat darurat	1.902636895317040
6	RSU Islam Aisyiyah Malang	-7.98353070	112.62257480	STAND BY	jenazah	3.245554275468415
7	RS Umum Universitas Muhammadiyah Malang	-7.93543200	112.70675080	STAND BY	transportasi	10.88499744671158
8	RS Umum Lavalette Malang	-7.96584900	112.63797200	STAND BY	gawat darurat	3.199335097591052
9	RS Umum Panti Waluya Sawahan Malang	-7.98656170	112.62277880	STAND BY	gawat darurat	3.199335097591052
10	RS Umum Panti Nirmala	-7.91561710	112.60439780	STAND BY	transportasi	4.645464456063804
11	RS Umum Daerah Dr. Saiful Anwar	-7.97256200	112.62935690	STAND BY	jenazah	2.718550357889309

12	Puskesmas Gribig	-7.98068070	112.66311030	STAND BY	transportasi	6.381375892717393
13	Puskesmas Arjuno	-7.97813340	112.62444200	OFF DUTY	transportasi	2.818972351170904
14	Puskesmas Ciptomulyo	-8.00200850	112.62886360	STAND BY	gawat darurat	5.406828581900004
15	Puskesmas Kendalkerep	-7.96123480	112.64873620	STAND BY	gawat darurat	4.255315311718729

6.1.9 Pengujian Kode KF-02-05

Pengujian kode KF-02-05 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.10.

Tabel 6. 10 Tabel Pengujian Fungsional KF-02-05

Kode Fungsi	KF-02-05
Kebutuhan Fungsional	Node *subscriber* dapat menerima *request* dari node aplikasi perangkat bergerak pasien untuk memberikan *response* berupa ambulans yang sesuai dengan lokasi pasien dan kasus pasien
Skenarion Pengujian	1. Node *subscriber* menerima *request* dari node aplikasi perangkat bergerak pasien 2. Menampilkan *request* yang diterima dan *response* yang dikirimkan
Hasil yang Diharapkan	Node *subscriber* dapat menerima *request* dan menampilkan *request* yang diterima serta menampilkan *response* yang dikirimkan
Hasil Pengujian	Node *subscriber* berhasil menerima *request* dan menampilkan *request* yang diterima serta menampilkan response yang dikirimkan

Hasil pengujian kode KF-02-05 adalah Node *subscriber* berhasil menerima *request* dan menampilkan *request* yang diterima serta menampilkan *response* yang dikirimkan. Pada Gambar 6.17 adalah terminal node *subscriber* ketika mendapatkan *request* dengan method POST. Dan Gambar 6.18 adalah terminal node *subscriber* ketika melakukan *response* terhadapt *request* POST tersebut. *Response* yang dikirimkan merupakan string dengan format JSON yang berisi keterangan apakah *request* berhasil dilakukan dan data ambulans yang dibutuhkan oleh pasien.

```
 * Debug mode: off
[2019-11-19 17:24:52,050] DEBUG in __init__: Subscribed to topic: TestingTopic, qos: 0
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
Ambulans :
{'id': 1, 'ket': 'RS Universitas Brawijaya', 'lat': '-7.94172720', 'lon': '112.62187210', '
jenis': 'STAND BY', 'jarak': '2.1190731546155206'}
Response : {"username": "farah", "success": "1", "message": "success", "ambulans": {"id": 1
, "ket": "RS Universitas Brawijaya", "lat": "-7.94172720", "lon": "112.62187210", "jenis":
"STAND BY", "jarak": "2.1190731546155206"}}
192.168.1.8 - - [19/Nov/2019 17:25:11] "POST /getAmbulans HTTP/1.1" 200 -
```

Gambar 6. 17 Terminal Node *Subscriber* Menampilkan *Request* yang Diterima

```
   Use a production WSGI server instead.
 * Debug mode: off
[2019-11-19 17:24:52,050] DEBUG in __init__: Subscribed to topic: TestingTopic, qos: 0
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
Ambulans :
{'id': 1, 'ket': 'RS Universitas Brawijaya', 'lat': '-7.94172720', 'lon': '112.62187210', '
jenis': 'STAND BY', 'jarak': '2.1190731546155206'}
Response : {"username": "farah", "success": "1", "message": "success", "ambulans": {"id": 1
, "ket": "RS Universitas Brawijaya", "lat": "-7.94172720", "lon": "112.62187210", "jenis":
"STAND BY", "jarak": "2.1190731546155206"}}
192.168.1.8 - - [19/Nov/2019 17:25:11] "POST /getAmbulans HTTP/1.1" 200 -
```

Gambar 6. 18 Terminal Node *Subscriber* Menampilkan *Response* yang Dikirimkan

6.1.10 Pengujian Kode KF-02-06

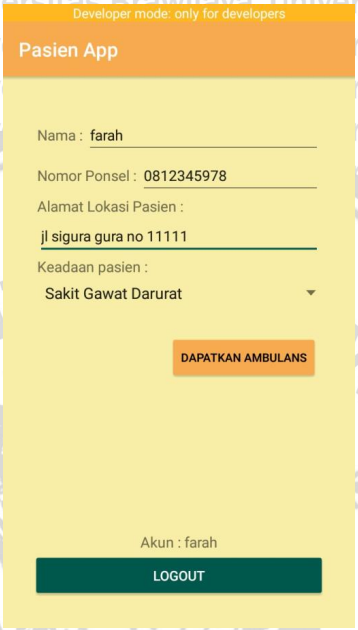
Pengujian kode KF-02-06 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.11.

Tabel 6. 11 Tabel Pengujian Fungsional KF-02-06

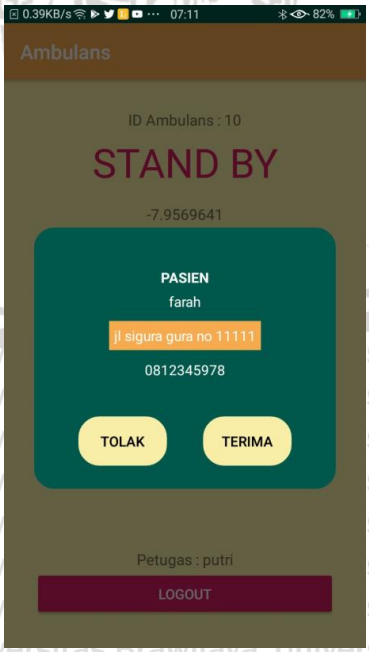
Kode Fungsi	KF-02-06
Kebutuhan Fungsional	Node *subscriber* dapat mengirimkan informasi pasien ke node *publisher*
Skenarion Pengujian	1. Node *subscriber* menerima request dari node aplikasi perangkat bergerak pasien 2. Node *subscriber* menentukan ambulans untuk pasien 3. Node *subscriber* mengirimkan informasi pasein ke node *publisher* dengan publisher yang memiliki ID ambulans yang sama dengan ID ambulans yang telah ditentukan sebelumnya 4. Node *publisher* menampilkan infromasi yang diterima oleh node *subscriber*
Hasil yang Diharapkan	Node subscriber webserver dapat mengirimkan informasi pasien ke node publisher dan node publisher dapat menampilkan informasi yang diterima
Hasil Pengujian	Node subscriber webserver berhasil mengirimkan informasi pasien ke node publisher dan node publisher berhasil

	menampilkan informasi yang diterima

Hasil pengujian kode KF-02-05 adalah Node subscriber webserver berhasil mengirimkan informasi pasien ke node publisher dan node publisher berhasil menampilkan informasi yang diterima. Pada Gambar 6.19 adalah Aplikasi pasien yang akan melakukan permintaan ambulans. Gambar 6.20 adalah aplikasi publisher yang menerima permintaan ambulans dan menampilkan informasi pasein.



Gambar 6. 19 Aplikasi Pasien Mengisi Data Pasien



Gambar 6. 20 Aplikasi Publisher Menerima Informasi Pasien

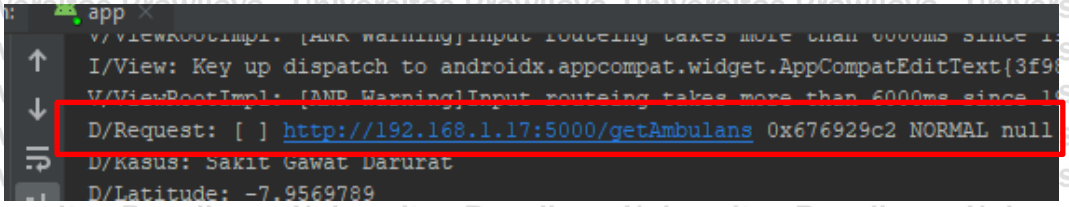
6.1.11 Pengujian Kode KF-03-01

Pengujian kode KF-03-01 dilakukan dengan skenario pengujian dan memiliki hasil pengujian seperti yang ditunjukan pada Tabel 6.12.

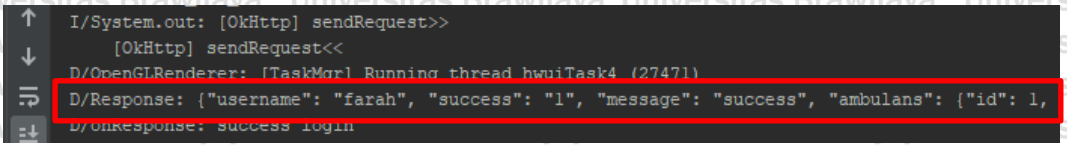
Tabel 6. 12 Tabel Pengujian Fungsional KF-03-01

Kode Fungsi	KF-03-01
Kebutuhan Fungsional	Node aplikasi perangkat bergerak pasien dapat mengirimkan *request* ke node *subscriber* untuk mendapatkan informasi mengenai ambulans yang sesuai dengan lokasi pasien dan kasus pasien
Skenarion Pengujian	1. Menyalakan *location service* pada perangkat 2. Node aplikasi perangkat bergerak pasein mengimplementasikan Google Play Service Location API untuk mendapatkan lokasi perangkat dalam bentuk koordinat 3. Node aplikasi pasien menampilkan form untuk menerima *input* dari *user* 4. Node aplikasi pasien mengirimkan *request* ke *subscriber*/webserver 5. Menampilkan *request* yang dikirimkan dan *response* yang diterima
Hasil yang Diharapkan	Node aplikasi perangkat bergerak pasien dapat mengirimkan *request* dan menampilkan *request* yang dikirimkan serta menampilkan *response* yang diterima
Hasil Pengujian	Node aplikasi perangkat bergerak pasien berhasil mengirimkan *request* dan menampilkan *request* yang dikirimkan serta menampilkan *response* yang diterima

Hasil pengujian kode KF-03-01 adalah Node aplikasi perangkat bergerak pasien berhasil mengirimkan *request* dan menampilkan *request* yang dikirimkan serta menampilkan *response* yang diterima. Pada Gambar 6.21 pesan log aplikasi pasien ketika mengirimkan *request* ke web server dengan method POST. Gambar 6.22 dan Gambar 6.23 adalah pesan log aplikasi pasien ketika mendapatkan *response* dari web server. *Response* yang didapatkan merupakan string dengan format JSON yang berisi data ambulans yang dibutuhkan oleh pasien.



Gambar 6. 21 Pesan Log Aplikasi Pasien Menampilkan *Request* yang Dikirimkan

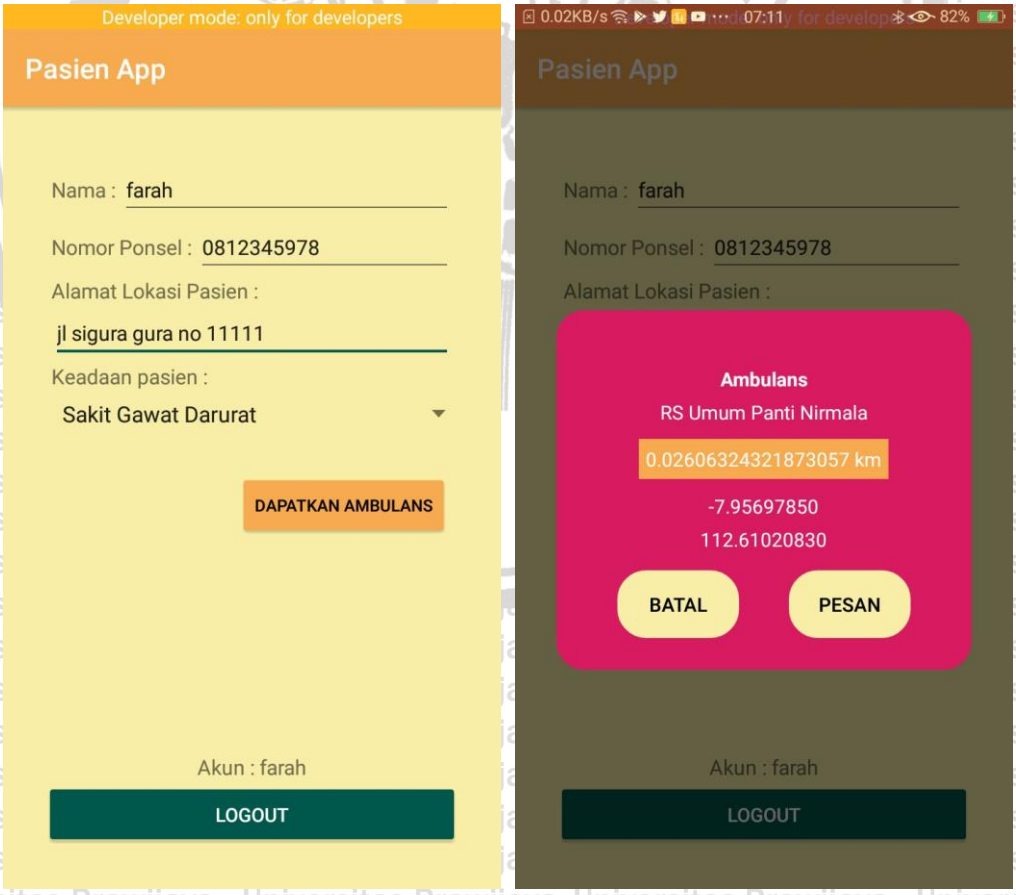


Gambar 6. 22 Pesan Log Aplikasi Pasien Menampilkan *Response* yang Diterima

Gambar 6. 23 Pesan Log Aplikasi Pasien Menampilkan *Response* yang Diterima (lanjutan)

Gambar 6.24 adalah tampilan aplikasi perangkat bergerak pasien untuk mendapatkan ambulans dan tampilan aplikasi perangkat bergerak pasien ketika mendapatkan ambulans yang dibutuhkan oleh pasien tersebut.

Ketika *user* telah mengisi *form* dan memilih tombol dapatkan ambulans, maka aplikasi perangkat bergerak pasien akan mengirimkan *request* ke web server. Web server akan mengirimkan *response* yang akan diterima oleh aplikasi perangkat bergerak pasien. *Response* data ambulans yang didapatkan akan ditampilkan pada halaman utama.



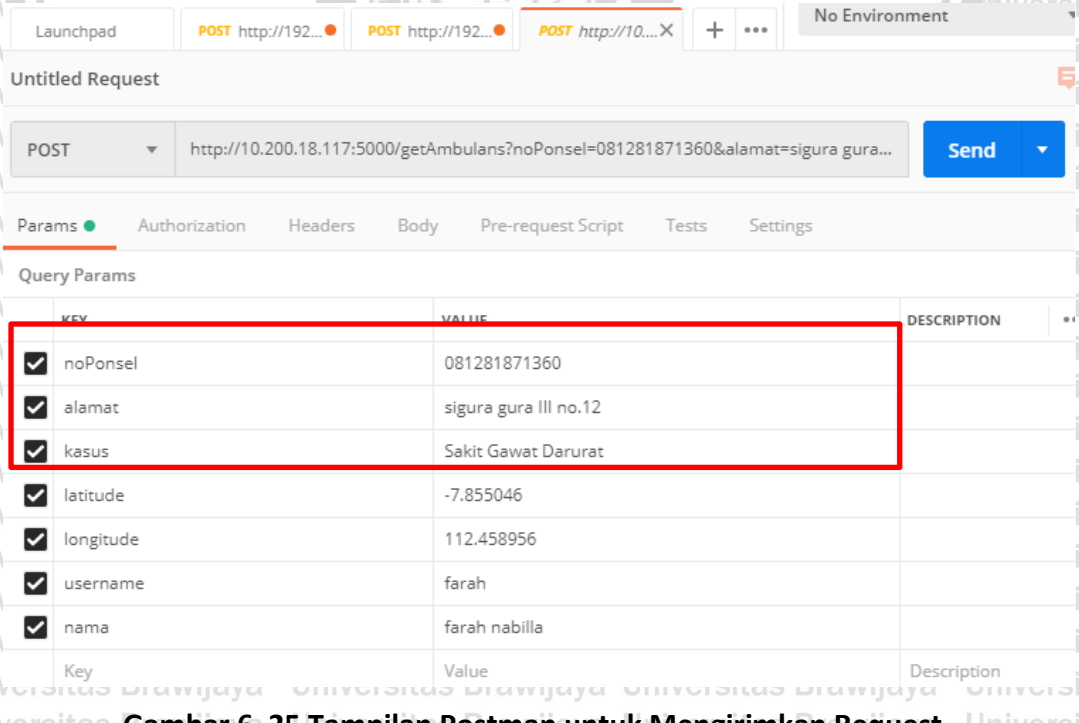
Gambar 6. 24 Tampilan Aplikasi Perangkat Bergerak Pasien

6.2 Pengujian Penentuan Ambulans

Pengujian penentuan ambulans dilakukan untuk mengetahui apakah penentuan ambulans menghasilkan hasil yang benar. Penentuan ambulans dilakukan dengan menentukan jenis ambulans berdasarkan kasus pasien. Kasus pasien sakit gawat darurat akan mendapatkan ambulans dengan jenis gawat darurat, kasus pasien sakit tidak gawat darurat akan mendapatkan ambulans dengan jenis transportasi dan pasien jenazah akan mendapatkan ambulans jenazah. Setelah diketahui jenis ambulans, akan mengambil data ambulans pada basis data berdasarkan jenis yang telah diketahui dan status STAND BY. Status STAND BY berarti ambulans siap untuk digunakan. Dari data yang didapatkan pada basis data, diketahui titik koordinat ambulans yang dapat digunakan menghitung jarak antar ambulans dan pasien. Dengan diketahuinya jarak antar ambulans dan pasien dapat dilakukan perbandingan jarak untuk menemukan ambulans dengan jarak terdekat.

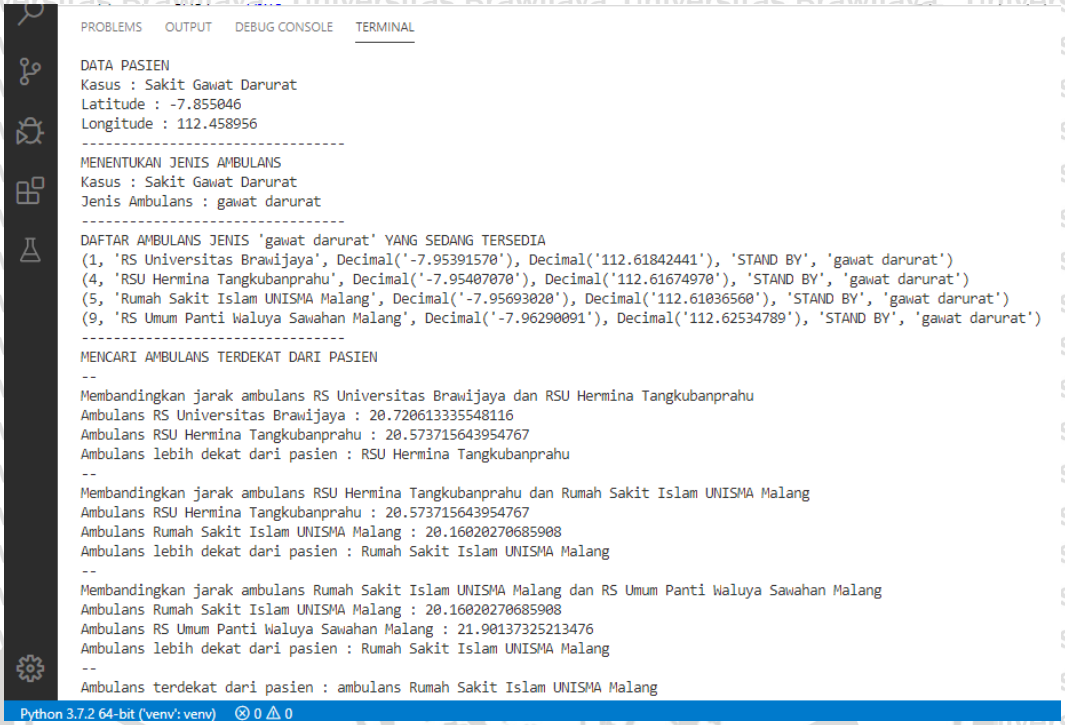
6.2.1 Jenis Ambulans Gawat Darurat

Pengujian ini untuk menguji penentuan ambulans terhadap ambulans jenis gawat darurat. Ambulans Jenis gawat darurat dipilih untuk menjemput pasien dengan keadaan gawat darurat. Gambar 6.25 Menunjukan request yang dikirimkan untuk mendapatkan ambulans. Pada request tersebut dikirimkan data kasus, latitude dan longitude. Latitude dan longitude diperlukan untuk mengetahui lokasi pasien.



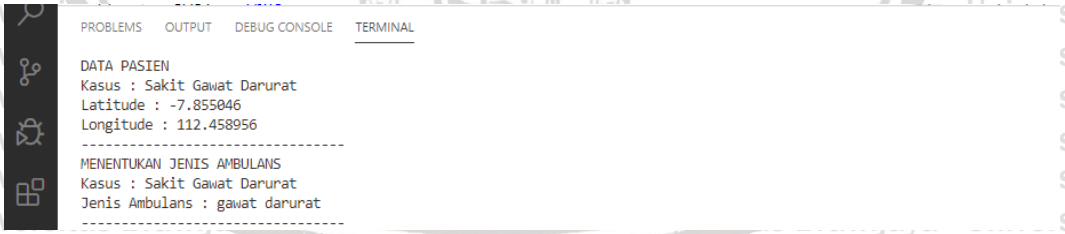
Gambar 6. 25 Tampilan Postman untuk Mengirimkan Request

Gambar 6.26 merupakan proses penentuan ambulans yang ditampilkan pada terminal web server.



Gambar 6. 26 Tampilan Terminal pada Web Server Ketika Menentukan Ambulans

Pertama dilakukan penentuan jenis ambulans berdasarkan kasus pasien. Kasus pasien yang diterima merupakan pasien sakit gawat darurat. Pasien sakit gawat darurat akan mendapatkan jenis ambulans gawat darurat seperti yang ditunjukan pada Gambar 6.27.



Gambar 6. 27 Penentuan Jenis Ambulans

Setelah mengetahui jenis ambulans, maka akan mengambil data ambulans pada basis data dengan jenis ambulans gawat darurat dan status ambulans STAND BY. Ambulans yang dapat digunakan oleh pasien hanya ambulans dengan status STAND BY yang berarti ambulans siap untuk digunakan. Oleh karena itu ambulans yang memenuhi kasus pasien dan tersedia adalah ambulans dengan id 1, 4, 5, dan 9. Gambar 6.28 adalah data ambulans yang ada pada basis data.

ID	hospital	latitude	longitude	status	jenis
1	RS Universitas Brawijaya	-7.95391570	112.61842441	STAND BY	gawat darurat
2	RS Umum Daerah Kota Malang	-7.95695120	112.61034720	ON DUTY	gawat darurat
3	RSU Persada Hospital	-7.95698990	112.61035340	OFF DUTY	gawat darurat
4	RSU Hermina Tangkubanprahu	-7.95407070	112.61674970	STAND BY	gawat darurat
5	Rumah Sakit Islam UNISMA Malang	-7.95693020	112.61036560	STAND BY	gawat darurat
6	RSU Islam Aisyiyah Malang	-7.95457671	112.71670479	STAND BY	transportasi
7	RS Umum Universitas Muhammadiyah Malang	-7.95407070	112.61674970	OFF DUTY	transportasi
8	RS Umum Lavalette Malang	-7.89209173	112.61379164	STAND BY	jenazah
9	RS Umum Panti Waluya Sawahan Malang	-7.96290091	112.62534789	STAND BY	gawat darurat
10	RS Umum Panti Nirmala	-7.93407573	112.69457543	STAND BY	jenazah
12	RSU Pacitan	-7.12275642	112.26881897	STAND BY	transportasi
13	RSU Ponorogo	-7.87609483	112.21859442	ON DUTY	transportasi
14	RS Aisyiah	-7.43617464	112.63885437	OFF DUTY	jenazah
15	RSI A Darmayu	-7.52897046	112.65868194	STAND BY	jenazah
16	RSU Dr Soetomo Trenggalek	-7.35003897	112.67535023	STAND BY	transportasi

Gambar 6. 28 Basis Data Ambulans

Gambar 6.29 Merupakan tampilan terminal pada web server yang berhasil mengambilan data pada basis data berdasarkan jenis ambulans dan status ambulans yang dibutuhkan.

```
--------------------------------
DAFTAR AMBULANS JENIS 'gawat darurat' YANG SEDANG TERSEDIA
(1, 'RS Universitas Brawijaya', Decimal('-7.95391570'), Decimal('112.61842441'), 'STAND BY', 'gawat darurat')
(4, 'RSU Hermina Tangkubanprahu', Decimal('-7.95407070'), Decimal('112.61674970'), 'STAND BY', 'gawat darurat')
(5, 'Rumah Sakit Islam UNISMA Malang', Decimal('-7.95693020'), Decimal('112.61036560'), 'STAND BY', 'gawat darurat')
(9, 'RS Umum Panti Waluya Sawahan Malang', Decimal('-7.96290091'), Decimal('112.62534789'), 'STAND BY', 'gawat darurat')
--------------------------------
```

Gambar 6. 29 Data Ambulans Gawat Darurat

Setelah diketahui ambulans-ambulans yang memenuhi, maka dihitung jarak ambulans tersebut dengan lokasi pasien dan dibandingkan antar ambulans tersebut untuk mengetahui ambulans terdekat. Perbandingan ambulans ditampilkan di terminal web server pada Gambar 6.30 Terlihat bahwa ambulans dari Rumah Sakit Islam UNISMA Malang merupakan ambulans terdekat.

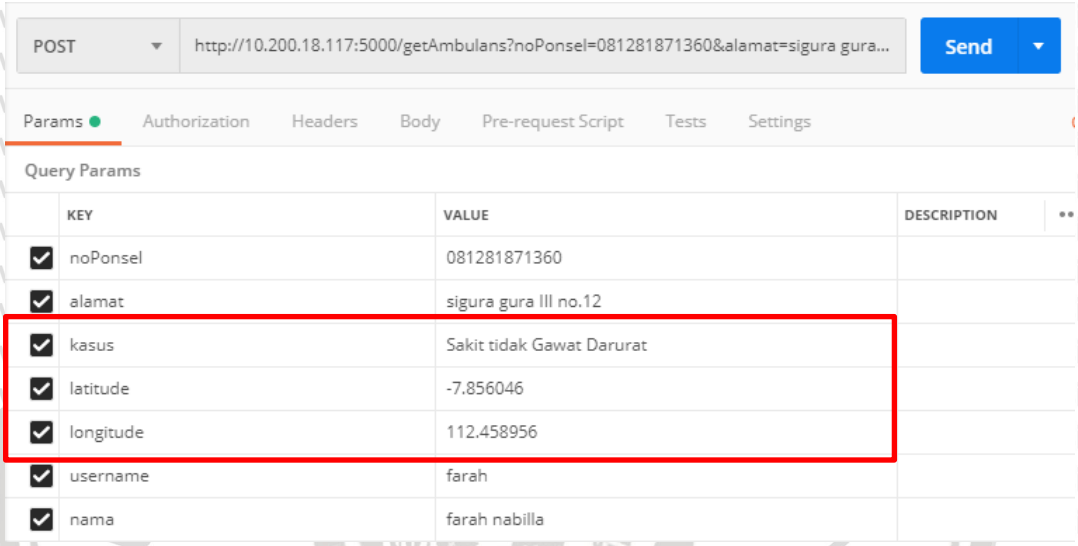
```
--------------------------------
MENCARI AMBULANS TERDEKAT DARI PASIEN
--
Membandingkan jarak ambulans RS Universitas Brawijaya dan RSU Hermina Tangkubanprahu
Ambulans RS Universitas Brawijaya : 20.720613335548116
Ambulans RSU Hermina Tangkubanprahu : 20.573715643954767
Ambulans lebih dekat dari pasien : RSU Hermina Tangkubanprahu
--
Membandingkan jarak ambulans RSU Hermina Tangkubanprahu dan Rumah Sakit Islam UNISMA Malang
Ambulans RSU Hermina Tangkubanprahu : 20.573715643954767
Ambulans Rumah Sakit Islam UNISMA Malang : 20.16020270685908
Ambulans lebih dekat dari pasien : Rumah Sakit Islam UNISMA Malang
--
Membandingkan jarak ambulans Rumah Sakit Islam UNISMA Malang dan RS Umum Panti Waluya Sawahan Malang
Ambulans Rumah Sakit Islam UNISMA Malang : 20.16020270685908
Ambulans RS Umum Panti Waluya Sawahan Malang : 21.90137325213476
Ambulans lebih dekat dari pasien : Rumah Sakit Islam UNISMA Malang

Ambulans terdekat dari pasien : ambulans Rumah Sakit Islam UNISMA Malang
```

Gambar 6. 30 Ambulans Gawat Darurat Terdekat

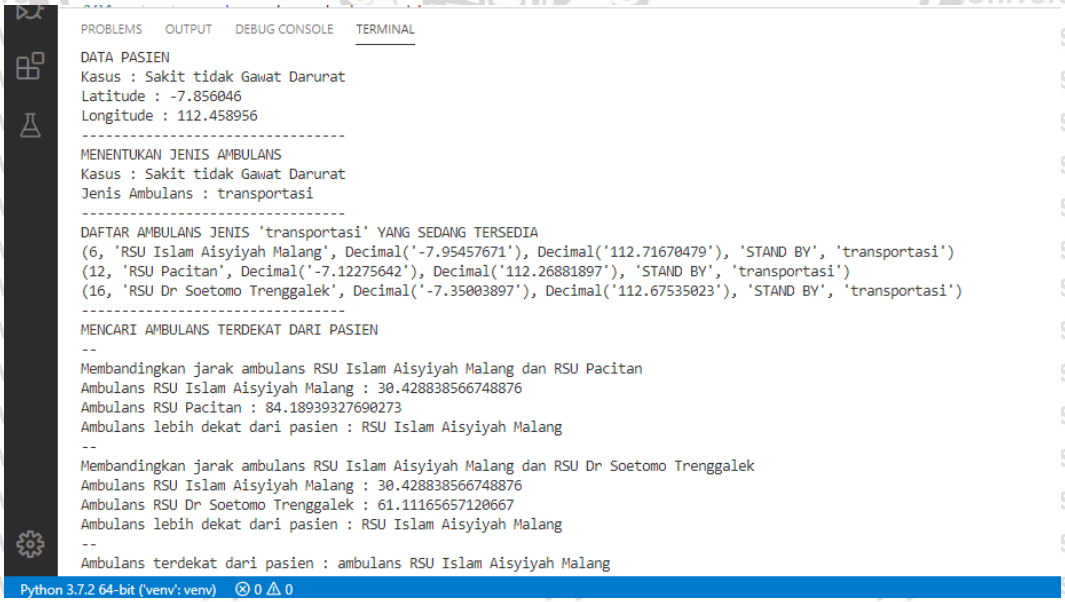
6.2.2 Jenis Ambulans Transportasi

Pengujian ini untuk menguji penentuan ambulans terhadap ambulans jenis transportasi. Ambulans Jenis transportasi dipilih untuk menjemput pasien dengan keadaan tidak gawat darurat. Gambar 6.31 Menunjukan request yang dikirimkan untuk mendapatkan ambulans. Pada request tersebut dikirimkan data kasus, latitude dan longitude. Latitude dan longitude diperlukan untuk mengetahui lokasi pasien.



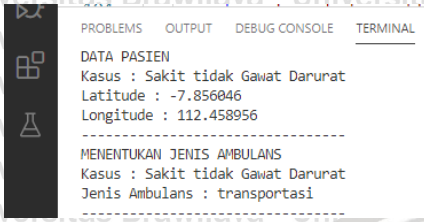
Gambar 6. 31 Tampilan Postman untuk Mengirimkan Request

Gambar 6.32 merupakan proses penentuan ambulans yang ditampilkan pada terminal web server.



Gambar 6. 32 Tampilan Terminal pada Web Server Ketika Menentukan Ambulans

Pertama dilakukan penentuan jenis ambulans berdasarkan kasus pasien. Kasus pasien yang diterima merupakan pasien sakit tidak gawat darurat. Pasien sakit tidak gawat darurat akan mendapatkan jenis ambulans transportadi seperti yang ditunjukan pada Gambar 6.33.



Gambar 6. 33 Penentuan Jenis Ambulans

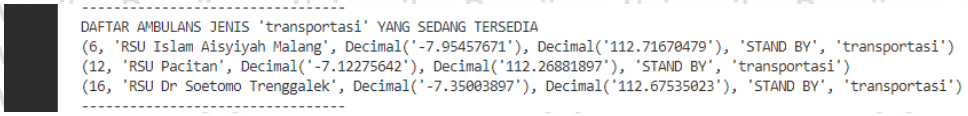
Setelah mengetahui jenis ambulans, maka akan mengambil data ambulans pada basis data dengan jenis ambulans transportasi dan status ambulans STAND BY. Ambulans yang dapat digunakan oleh pasien hanya ambulans dengan status STAND BY yang berarti ambulans siap untuk digunakan. Oleh karena itu ambulans yang memenuhi kasus pasien dan tersedia adalah ambulans dengan id 6, 12, dan 16. Gambar 6.34 adalah data ambulans yang ada pada basis data.

SQL | Search | Insert | Export | Import | Privileges | Operations

ID ▲ 1	hospital	latitude	longitude	status	jenis
1	RS Universitas Brawijaya	-7.95391570	112.61842441	STAND BY	gawat darurat
2	RS Umum Daerah Kota Malang	-7.95695120	112.61034720	ON DUTY	gawat darurat
3	RSU Persada Hospital	-7.95698990	112.61035340	OFF DUTY	gawat darurat
4	RSU Hermina Tangkubanprahu	-7.95407070	112.61674970	STAND BY	gawat darurat
5	Rumah Sakit Islam UNISMA Malang	-7.95693020	112.61036560	STAND BY	gawat darurat
6	RSU Islam Aisyiyah Malang	-7.95457671	112.71670479	STAND BY	transportasi
7	RS Umum Universitas Muhammadiyah Malang	-7.95407070	112.61674970	OFF DUTY	transportasi
8	RS Umum Lavalette Malang	-7.89209173	112.61379164	STAND BY	jenazah
9	RS Umum Panti Waluya Sawahan Malang	-7.96290091	112.62534789	STAND BY	gawat darurat
10	RS Umum Panti Nirmala	-7.93407573	112.69457543	STAND BY	jenazah
12	RSU Pacitan	-7.12275642	112.26881897	STAND BY	transportasi
13	RSU Ponorogo	-7.87609483	112.21859442	ON DUTY	transportasi
14	RS Aisyiah	-7.43617464	112.63885437	OFF DUTY	jenazah
15	RSI A Darmayu	-7.52897046	112.65868194	STAND BY	jenazah
16	RSU Dr Soetomo Trenggalek	-7.35003897	112.67535023	STAND BY	transportasi

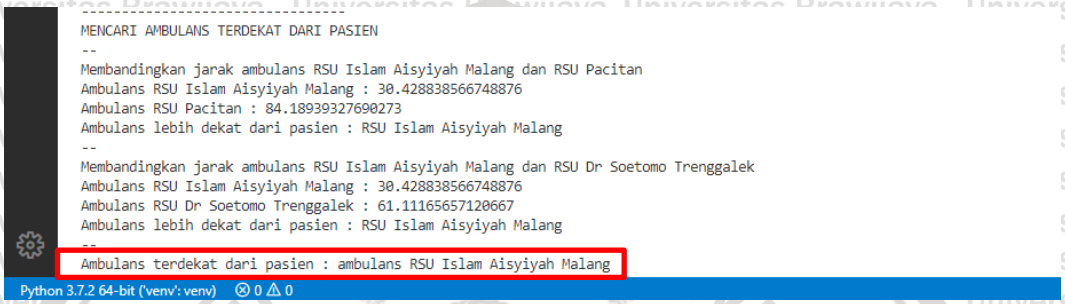
Gambar 6. 34 Basis Data Ambulans

Gambar 6.35 Merupakan tampilan terminal pada web server yang berhasil mengambilan data pada basis data berdasarkan jenis ambulans dan status ambulans yang dibutuhkan.



Gambar 6. 35 Data Ambulans Transportasi

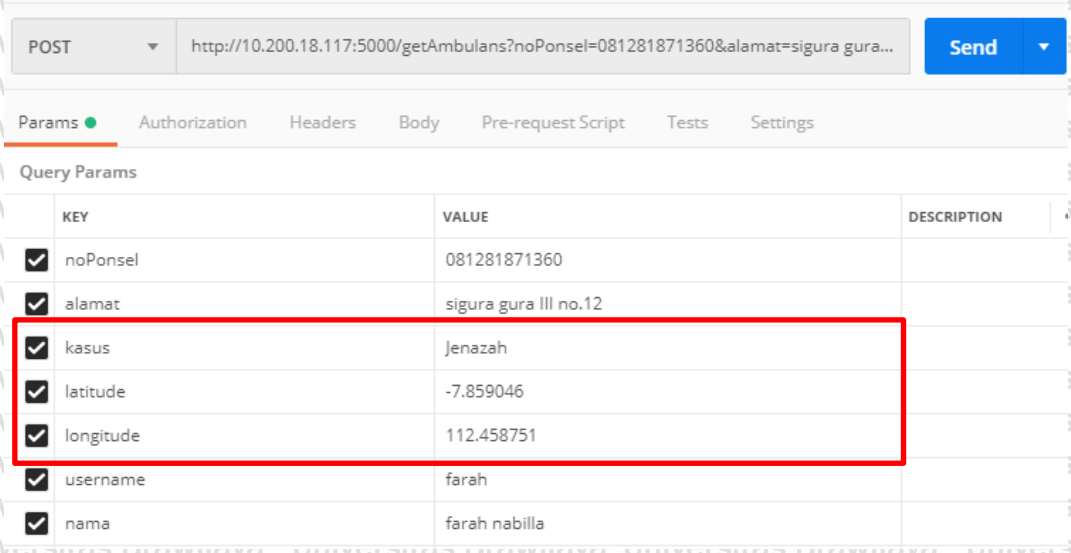
Setelah diketahui ambulans-ambulans yang memenuhi, maka dihitung jarak ambulans tersebut dengan lokasi pasien dan dibandingkan antar ambulans tersebut untuk mengetahui ambulans terdekat. Perbandingan ambulans ditampilkan di terminal web server pada Gambar 6.36 Terlihat bahwa ambulans dari RSU Islam Aisyiyah Malang merupakan ambulans terdekat.



Gambar 6. 36 Ambulans Transportasi Terdekat

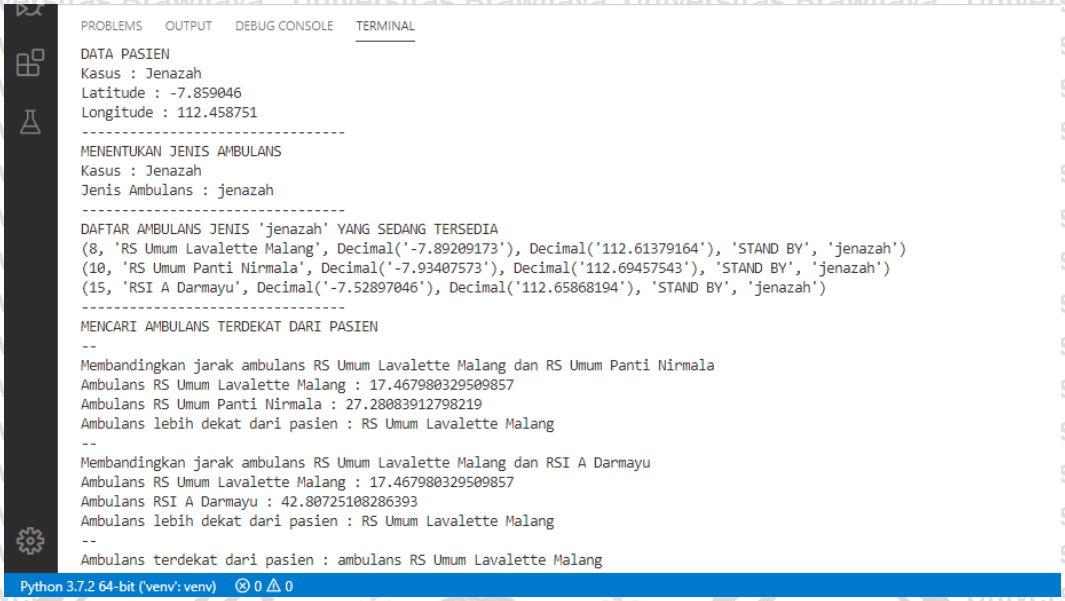
6.2.3 Jenis Ambulans Jenazah

Pengujian ini untuk menguji penentuan ambulans terhadap ambulans jenis jenazah. Ambulans Jenis jenazah dipilih untuk menjemput jenazah. Gambar 6.37 Menunjukan request yang dikirimkan untuk mendapatkan ambulans. Pada request tersebut dikirimkan data kasus, latitude dan longitude. Latitude dan longitude diperlukan untuk mengetahui lokasi pasien.



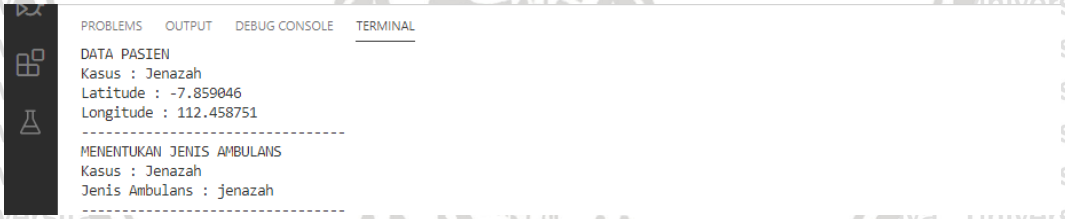
Gambar 6. 37 Tampilan Postman untuk Mengirimkan Request

Gambar 6.38 merupakan proses penentuan ambulans yang ditampilkan pada terminal web server.



Gambar 6. 38 Tampilan Terminal pada Web Server Ketika Menentukan Ambulans

Pertama dilakukan penentuan jenis ambulans berdasarkan kasus pasien. Kasus pasien yang diterima merupakan jenazah. Jenazah akan mendapatkan jenis ambulans jenazah seperti yang ditunjukan pada Gambar 6.39



Gambar 6. 39 Penentuan Jenis Ambulans

Setelah mengetahui jenis ambulans, maka akan mengambil data ambulans pada basis data dengan jenis ambulans jenazah dan status ambulans STAND BY. Ambulans yang dapat digunakan hanya ambulans dengan status STAND BY yang berarti ambulans siap untuk digunakan. Oleh karena itu ambulans yang memenuhi kasus pasien dan tersedia adalah ambulans dengan id 8, 10, dan 15. Gambar 6.40 adalah data ambulans yang ada pada basis data.

SQL | Search | Insert | Export | Import | Privileges | Operations

ID	hospital	latitude	longitude	status	jenis
1	RS Universitas Brawijaya	-7.95391570	112.61842441	STAND BY	gawat darurat
2	RS Umum Daerah Kota Malang	-7.95695120	112.61034720	ON DUTY	gawat darurat
3	RSU Persada Hospital	-7.95698990	112.61035340	OFF DUTY	gawat darurat
4	RSU Hermina Tangkubanprahu	-7.95407070	112.61674970	STAND BY	gawat darurat
5	Rumah Sakit Islam UNISMA Malang	-7.95693020	112.61036560	STAND BY	gawat darurat
6	RSU Islam Aisyiyah Malang	-7.95457671	112.71670479	STAND BY	transportasi
7	RS Umum Universitas Muhammadiyah Malang	-7.95407070	112.61674970	OFF DUTY	transportasi
8	RS Umum Lavalette Malang	-7.89209173	112.61379164	STAND BY	jenazah
9	RS Umum Panti Waluya Sawahan Malang	-7.96290091	112.62534789	STAND BY	gawat darurat
10	RS Umum Panti Nirmala	-7.93407573	112.69457543	STAND BY	jenazah
12	RSU Pacitan	-7.12275642	112.26881897	STAND BY	transportasi
13	RSU Ponorogo	-7.87609483	112.21859442	ON DUTY	transportasi
14	RS Aisyiah	-7.43617464	112.63885437	OFF DUTY	jenazah
15	RSI A Darmayu	-7.52897046	112.65868194	STAND BY	jenazah
16	RSU Dr Soetomo Trenggalek	-7.35003897	112.67535023	STAND BY	transportasi

Gambar 6. 40 Basis Data Ambulans

Gambar 6.41 Merupakan tampilan terminal pada web server yang berhasil mengambilan data pada basis data berdasarkan jenis ambulans dan status ambulans yang dibutuhkan.

```
---------------------------------
DAFTAR AMBULANS JENIS 'jenazah' YANG SEDANG TERSEDIA
(8, 'RS Umum Lavalette Malang', Decimal('-7.89209173'), Decimal('112.61379164'), 'STAND BY', 'jenazah')
(10, 'RS Umum Panti Nirmala', Decimal('-7.93407573'), Decimal('112.69457543'), 'STAND BY', 'jenazah')
(15, 'RSI A Darmayu', Decimal('-7.52897046'), Decimal('112.65868194'), 'STAND BY', 'jenazah')
---------------------------------
```

Gambar 6. 41 Data Ambulans Jenazah

Setelah diketahui ambulans-ambulans yang memenuhi, maka dihitung jarak ambulans tersebut dengan lokasi pasien dan dibandingkan antar ambulans tersebut untuk mengetahui ambulans terdekat. Perbandingan ambulans ditampilkan di terminal web server pada Gambar 6.42 Terlihat bahwa ambulans dari RS Umum Lavelette Malang merupakan ambulans terdekat.

```
MENCARI AMBULANS TERDEKAT DARI PASIEN
--
Membandingkan jarak ambulans RS Umum Lavalette Malang dan RS Umum Panti Nirmala
Ambulans RS Umum Lavalette Malang : 17.467980329509857
Ambulans RS Umum Panti Nirmala : 27.28083912798219
Ambulans lebih dekat dari pasien : RS Umum Lavalette Malang
--
Membandingkan jarak ambulans RS Umum Lavalette Malang dan RSI A Darmayu
Ambulans RS Umum Lavalette Malang : 17.467980329509857
Ambulans RSI A Darmayu : 42.80725108286393
Ambulans lebih dekat dari pasien : RS Umum Lavalette Malang

Ambulans terdekat dari pasien : ambulans RS Umum Lavalette Malang
```

Gambar 6. 42 Ambulans Jenazah Terdekat

6.3 Pengujian Akurasi

Pengujian Akurasi Sistem dilakukan untuk mengetahui tingkat akurasi implementasi formula haversine dalam menentukan jarak antara titik koordinat perangkat node *publisher* dan titik koordinat node aplikasi perangkat bergerak pasien. Nilai akurasi didapatkan dari perhitungan persentase galat (*error*) relatif. Nilai persentase galat relatif membutuhkan nilai sejati dan nilai pendekatan. Nilai sejati dalam penelitian ini adalah jarak dua titik yang di dapatkan dari perhitungan google maps API. Nilai pendekatan adalah nilai perhitungan jarak dua titik yang didapatkan dari perhitungan menggunakan formula haversine. Tabel 6.13 adalah kode program mendapatkan jarak dua titik menggunakan google maps API dan Tabel 6.14 merupakan kode program untuk menghitung jarak dua titik menggunakan formula haversine.

Tabel 6. 13 Kode Program mendapatkan Nilai Sejati

Fungsi jarak

```
1  function jarak(latitude1,longitude1,latitude2,longitude2) {
2      var                          distance                          =
   google.maps.geometry.spherical.computeDistanceBetween(new
   google.maps.LatLng(latitude1,            longitude1),                 new
   google.maps.LatLng(latitude2, longitude2));
3      distance = distance * 0.001;
4      console.log(distance);
5      document.getElementById("demo").innerHTML = distance;
6  }
```

Tabel 6. 14 Kode Program mendapatkan Nilai Pendekatan

Fungsi haversine

```
1  def haversine(lat1,lon1,lat2,lon2):
2      lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
3
4      dlat = (lat2 - lat1)
5      dlon = (lon2 - lon1)
6
7      a = sin(dlat / 2) ** 2 + cos(lat1) * cos(lat2) * sin(dlon / 2) **
   2
8      d= 2 * 6371 * asin(sqrt(a))
9
10     print("d = "+str(d))
```

Tabel 6.15 merupakan kode program untuk menghitung galat. Dimana x1 merupakan nilai sejati yaitu jarak dua titik menggunakan google API dan x2 merupakan nilai pendekatan yaitu jarak dua titik menggunakan formula haversine.

Tabel 6. 15 Kode Program menghitung Galat

Fungsi galat

```
1  def galat(x1,x2):
2      e = abs((x1-x2)/x1)
3      result = e * 100
4      print(result)
```

Tabel 6.16 adalah hasil pengujian akurasi. Pengujian akurasi dilakukan 10 kali dengan titik koordinat yang berbeda-beda. Rata-rata persentase galat adalah 0,1118978787072161 % dan persentase akurasi adalah 99,88810212129279 %.

Tabel 6. 16 Hasil Pengujian Akurasi

No	Koordinat Asal	Koordinat Tujuan	Jarak dengan Google Maps API (km)	Jarak dengan Formula Haversine (km)	Persentase Galat (%)	Persentase Akurasi (%)
1	-7.941030 112.621523	-7.944493 112.601577	2,232611046430 5517489	2,2301128020284 67	0,111897878767 51038	99,8881021212324 9
2	-7.940222 112.608833	-7.857177 112.659035	10,77508158000 4789217	10,763024492293 747	0,111897878651 95035	99,8881021213480 6
3	-7.918548 112.583217	-7.939438 112.602906	3,181229980394 5758292	3,1776702515240 824	0,111897878884 31321	99,8881021211157
4	-7.938198 112.587550	-7.938676 112.599110	1,275631722306 6595545	1,2742043174672 98	0,111897878862 76092	99,8881021211372 4
5	-7.925318 112.596301	-7.945648 112.554001	5,183826122429 0489437	5,1780255309591 61	0,111897878765 46155	99,8881021212345 4
6	-7.861617 112.597063	-7.920273 112.558910	7,767469730081 101531	7,7587780962318 71	0,111897878604 79905	99,8881021213952
7	-7.958200 112.592546	-7.697352 112.857559	41,19884394666 3707527	41,152743314255 744	0,111897878658 06512	99,8881021213419 3
8	-7.691081 112.721406	-7.687483 112.832840	12,29975849951 0622	12,285995330671 353	0,111897878643 84787	99,8881021213561 6
9	-8.216212 112.487881	-8.179790 112.445891	6,151720431276 53	6,1448367866190 94	0,111897878558 30215	99,8881021214417
10	-8.052109 112.640812	-8.098230 112.492910	17,09053300284 7287	17,071409058962 825	0,111897878675 15036	99,8881021213248 5
Rata-rata					0,111897878707 2161	99,8881021212927 9

6.4 Pengujian *Response Time*

Pengujian *Response Time* adalah pengujian waktu respons yang mengacu pada waktu yang diperlukan untuk suatu node sistem untuk menanggapi permintaan. Pengujian *response time* pada penelitian dibagi menjadi dua pengujian. Pengujian *response time* yang pertama adalah pengujian waktu yang diperlukan untuk mengirimkan data lokasi dari *publisher* ke *subscriber* dan pengujian *response* time yang kedua adalah waktu yang diperlukan untuk menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans yang ada pada basis data.

6.3.1 Pengujian *Response Time* dari *Publisher* ke *Subscriber*

Pengujian *response time* yang pertama adalah pengujian waktu yang diperlukan untuk mengirimkan data lokasi dari *publisher* ke *subscriber*. Waktu dihitung dari penentuan lokasi pada node *publisher* sampai data sampai pada *subscriber*. Pengujian *response time* dilakukan dengan mendapatkan waktu sebelum node publisher melakukan *publish* dan mendapatkan waktu ketika data diterima oleh node *subscriber*. Kemudian menghitung selisih waktu setelah data diterima oleh node *subscriber* dan waktu sebelum melakukan *publish* pada node *publisher*. Pengujian *response time* ini dilakukan sebanyak 10 kali percobaan.

Tabel 6.17 adalah kode program pada *publisher* ketika memilih tombol pilih. Pada tombol pilih akan melakukan perulangan untuk melakukan *publish* atau menghubungkan *publisher* ke *broker* apabila belum terhubung. Pada baris 10 merupakan kode program untuk mendapatkan waktu saat ini dalam satuan *millisecond*. Waktu didapatkan sebelum melakukan *publish*. Pada method publish di baris 14 akan melakukan pengambilan data lokasi dengan GPS dan melakukan *publish* data ke *broker*.

Tabel 6. 17 Kode Program Mendapatkan Waktu Saat Ini pada *Publisher*

Fungsi btnPilih_onClick

```
1  public void btnPilih_onClick(View view){
2      . . .
3      if(bgthread==null||bgthread.getState()==
4  Thread.State.TERMINATED){
5          Runnable runnable = new Runnable() {
6              @Override
7              public void run() {
8                  try{
9                      while(!stop){
10                         Log.d("Waktu",
   String.valueOf(System.currentTimeMillis()));
11                         handler.post(new Runnable() {
12                             public void run() {
13                                 if(client.isConnected()){
14                                     publish(id,status, client);
15                                 }
16                                 else{
17                                     connectToMqtt(client);
18                                 }
19                             }
20                         });
21                         Thread.sleep(1000);
22                     }
23                 }
24                 catch (InterruptedException e){
25                     e.printStackTrace();
26                 }
27             }
28         };
29         bgthread = new Thread(runnable);
30         bgthread.start();
31     }
32 }
```

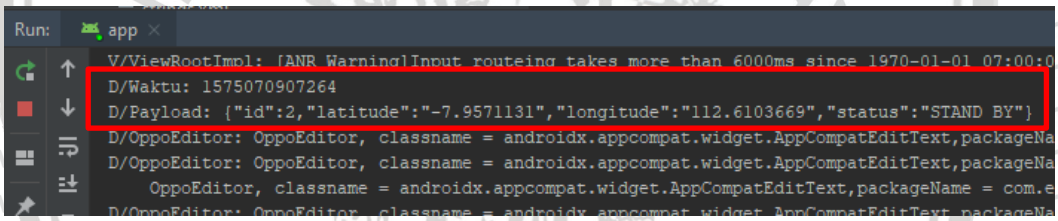
Tabel 6.18 adalah kode program *subscriber* ketika memilih mendapatkan pesan. Baris ke-5 merupakan kode program untuk mendapatkan waktu saat ini dalam satuan *millisecond*. Waktu didapatkan ketika menerima pesan.

Tabel 6. 18 Kode Program Mendapatkan Waktu Saat Ini pada *Subscriber*

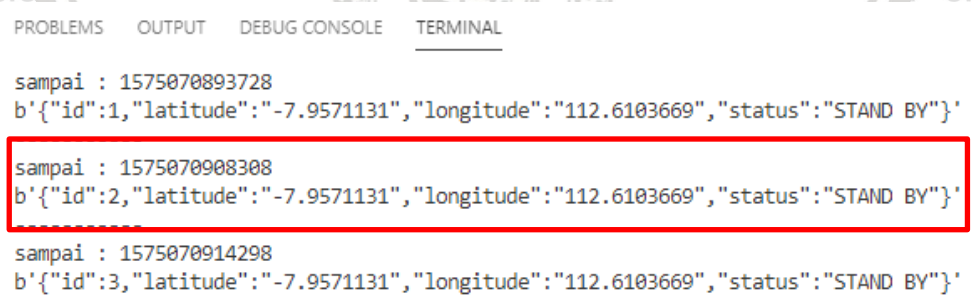
__init__.py

```
1  . . .
2
3  @mqtt.on_message()
4  def handle_mqtt_message(client, userdata, message):
5      milliseconds = int(round(time.time() * 1000))
6      print("sampai : "+str(milliseconds))
7      print(message.payload)
8      print("-----------")
9      . . .
```

Gambar 6.43 merupakan pesan log pada *publisher* untuk menampilkan waktu saat ini dalam satuan *milliseconds* dan Gambar 6.44 adalah terminal pada *subscriber* untuk menampilkan waktu saat ini dalam satuan *milliseconds*.



Gambar 6. 43 Log pada *Publisher* Menampilkan Waktu dalam *Millisecond*



Gambar 6. 44 Terminal pada *Subscriber* Menampilkan Waktu dalam *Milliseconds*

Tabel 6.19 merupakan hasil pengujian *response time* dari *publisher* ke *subscriber* dengan menghitung selisih waktu pada *publisher* sebelum menentukan lokasi dan mengirimkan data dengan waktu pada *subscriber* ketika menerima data. Dari tabel tersebut terlihat rata-rata *response time* pengiriman dara dari node *publisher* ke node *subscriber* adalah 880,2 *milliseconds*.

Tabel 6. 19 Hasil Pengujian *Response Time Publisher* ke *Subscriber*

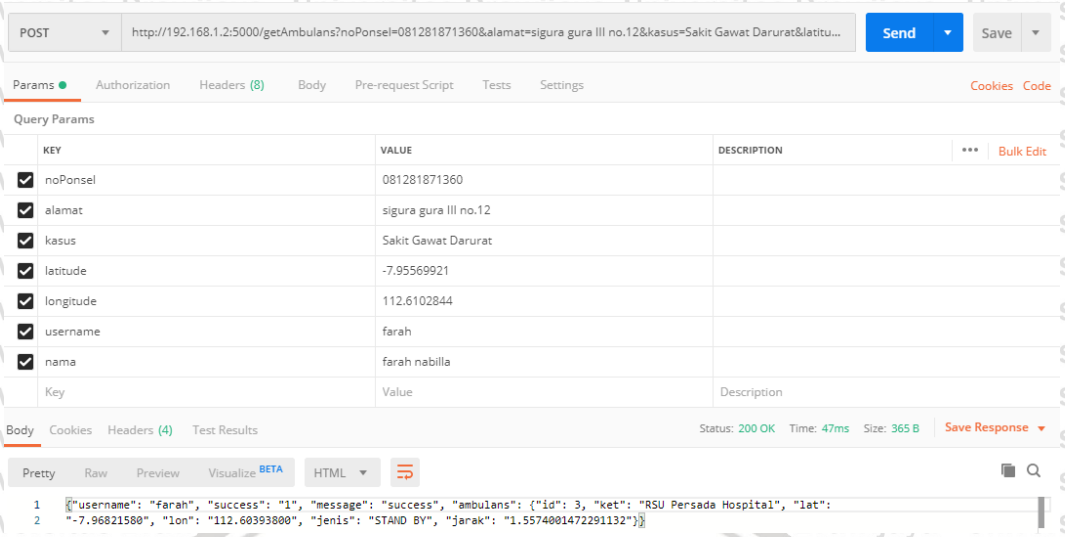
Percobaan	*Response Time* (*milliseconds*)
1	887
2	861
3	845
4	852
5	780
6	794
7	1172
8	677
9	737
10	1197
Rata-rata	880,2

6.3.2 Pengujian *Response Time* dari Node Aplikasi Pasien ke Node *Subscriber* (Web Server)

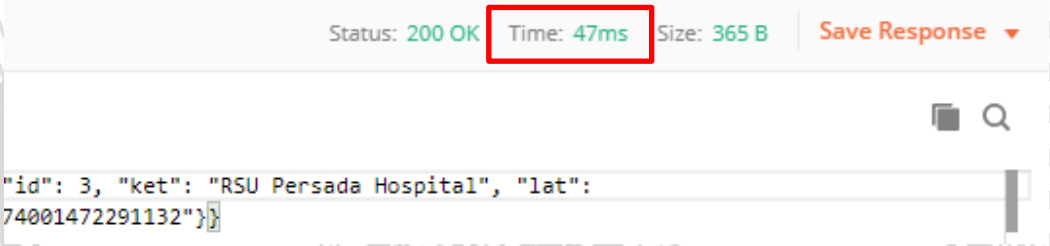
Pengujian *response time* ini menghitung waktu yang diperlukan dari aplikasi pasien ke node subscriber yang merupakan web server untuk menentukan ambulans dengan variasi jumlah data ambulans pada basis data. Variasi jumlah data pada basis data adalah 1000, 2000, 3000, 4000 dan 5000 data ambulans.

Pengujian ini dilakukan untuk mengetahui waktu respons menentukan ambulans dengan variasi jumlah data pada basis data yang berbeda-beda. Variasi jumlah ambulans pada basis data karena ambulans ditentukan dengan membandingkan seluruh data yang ada pada basis data untuk mencari ambulans terdekat dengan pasien.

Pengujian *response time* ini dilakukan dengan menggunakan alat bantu aplikasi postman. Postman adalah aplikasi yang berfungsi sebagai REST *Client* yang digunakan untuk melakukan uji coba REST API. Postman akan mengirimkan *request* ke web server dan menghitung *response time*. Pengujian dilakukan sebanyak 10 kali percobaan. Gambar 6.45 merupakan tampilan postman ketika mengirimkan *request* dan menerima *respons*. Gambar 6.46 menunjukan *response time* yang didapatkan.



Gambar 6. 45 Tampilan Postman



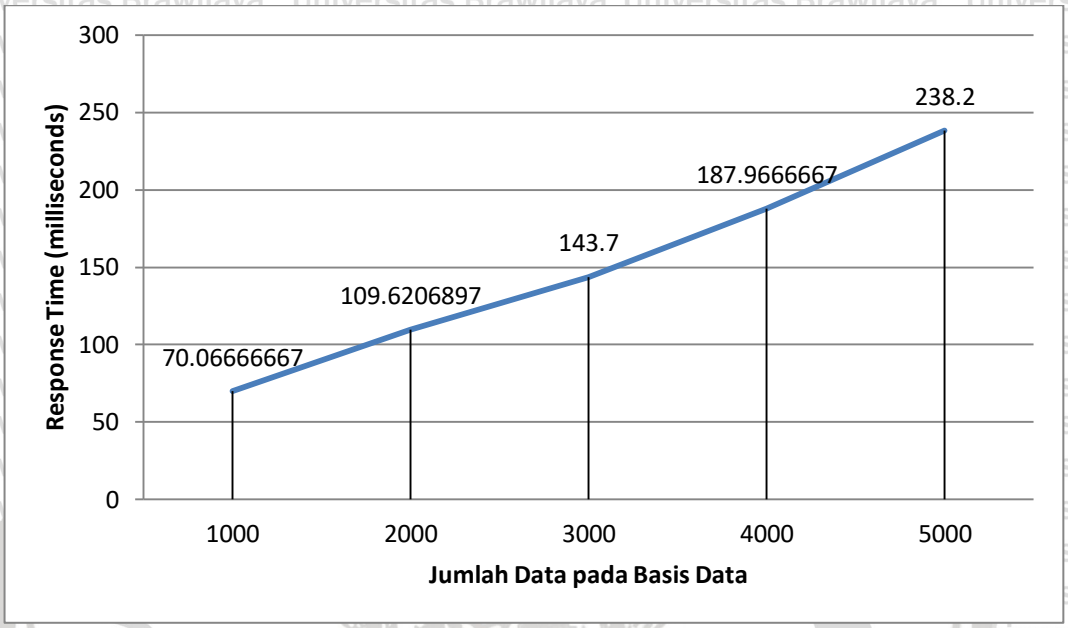
Gambar 6. 46 *Response Time* pada Postman

Pengujian response time ini dilakukan 30 kali untuk masing-masing variasi jumlah data pada basis data. Dari 30 kali percobaan tersebut akan dihitung rata-rata dari response time tersebut. Tabel 6.20 adalah hasil pengujian *response time* dari node aplikasi pasien ke node subscriber (web server). Rata-rata *response time* dengan 1000 data ambulans di basis data adalah 70,06666667 *seconds*. Rata-rata *response time* dengan 2000 data ambulans di basis data adalah 109,6206897. Rata-rata *response time* dengan 3000 data ambulans di basis data adalah 143,7. Rata-rata *response time* dengan 4000 data ambulans di basis data adalah 187,9666667. Rata-rata *response time* dengan 5000 data ambulans di basis data adalah 238,2.

Tabel 6. 20 Hasil Pengujian Response Time dari Aplikasi Pasien ke Node Subscriber

Jumlah Data Pada Basis Data	Response Time (*milliseconds*)
1000	70,06666667
2000	109,6206897
3000	143,7

4000	187,9666667
5000	238,2



Gambar 6. 47 Grafik *Response Time* Menentukan Ambulans

Hasil pengujian *response time* dari aplikasi pasien ke node *subscriber (*web server*)* untuk menentukan ambulans ditunjukan dalam bentuk grafik pada Gambar 6.47. Pada grafik tersebut terlihat bahwa semakin banyak data pada basis data ambulans, maka *response time* untuk menentukan ambulans akan semakin besar. Dari pengujian yang dilakukan, dapat disimpulkan bahwa banyak data ambulans pada basis data mempengaruhi *response time* aplikasi pasien ke node *subscriber* (web server) untuk menentukan ambulans.

BAB 7 PENUTUP

Bab ini memuat kesimpulan dan saran terhadap penelitian yang dilakukan. Kesimpulan dan saran didapatkan berdasarkan hasil perancangan, implementasi dan pengujian.

7.1 Kesimpulan

Kesimpulan yang didapatkan dari penelitian yang telah dilakukan adalah:

1. Metode komunikasi publish/subscribe untuk mengetahui posisi ambulansnnya berhasil diimplementasikan. Implementasi metode komunikasi publish/subscribe tersebut menggunakan protokol MQTT dimana aplikasi ambulans sebagai pengirim data (publisher) dan web server sebagai penerima data (subscriber). Data yang dikirimkan merupakan data dalam format JSON.
2. Algoritma pemilihan ambulans terdekat dari posisi pasien dilakukan dengan menentukan ambulans terdekat dari pasien berdasarkan kasus pasien dan status ambulans. Kasus pasien digunakan untuk menentukan jenis ambulans yang dibutuhkan oleh pasien dan status ambulans digunakan untuk mengetahui ketersediaan ambulans. Jarak ambulans dan pasien dihitung menggunakan formula haversine. Jarak ambulans-ambulans tersebut kemudian dibandingkan dengan satu sama lain untuk menemukan ambulans terdekat.
3. Sistem *tracking* posisi ambulans menggunakan formula haversine untuk menentukan jarak antara dua titik koordinat. Akurasi dari penentuan jarak antara dua titik koordinat dilakukan dengan membandingkan jarak yang didapatkan oleh perhitungan sistem *tracking* ambulans dengan jarak yang didapatkan oleh perhitungan Google Maps API. Rata-rata akurasi dari formula haversine yang digunakan pada sistem tracking ambulans adalah 99,88810212129279% dan rata-rata galat 0,1118978787072161% .
4. Waktu respons pengiriman data *publisher* ke *subscriber* yang dihitung dari penentuan lokasi hingga data diterima oleh *subscriber* memiliki rata-rata waktu 880,2 *milliseconds*. Selanjutnya rata-rata waktu *response time* mengirimkan permintaan data ambulans yang dibutuhkan pasien pada aplikasi pasien ke web server (node *subscriber*) dengan variasi jumlah data ambulans pada basis data menunjukan bahwa semakin banyak data ambulans maka akan semakin besar *response time* nya. *Response time* dengan 1000 data ambulans pada basis data memiliki rata-rata waktu respons 70,06666667 *milliseconds*, 2000 data ambulans memiliki rata-rata waktu respons 109,6206897 *milliseconds*, 3000 data ambulans memiliki rata-rata waktu respons 143,7 *milliseconds*, 4000 data ambulans memiliki rata-rata waktu respons 187,9666667 *milliseconds*, dan 5000 data ambulans memiliki rata-rata waktu respons 238,2 *milliseconds*.

7.2 Saran

Sistem yang dibangun pada penelitian ini belum sempurna sehingga diperlukan adanya penelitian untuk pengembangan sistem selanjutnya. Adapun saran yang peneliti sarankan yaitu :

1. Menggunakan protokol selain MQTT dalam mengimplementasikan metode komunikasi *publish/subscribe* seperti AMQP (*Advanced Message Queueing Protocol*), XMPP (*Extensible Messaging and Presence Protocol*) atau protokol lainnya.
2. Menambahkan fitur lain pada sistem tracking implementasi posisi ambulans.

DAFTAR REFERENSI

Ansari, D. B., Rehmi, A. U. & Mughal, R. A., 2018. Internet of Things (IoT) Protocols: A Brief Exploration of MQTT and CoAP. *International Journal of Computer Applications,* Volume 179, pp. 9-14.

Banerjee, T. & Sahni, S., 2015. Pubsub: An Efficient Pubish/Subscribe System. *IEEE Transactions on Computers,* 1 April, Volume 64, pp. 1119-1132.

Dizdarevic, J., Carpio, F., Jukan, A. & Masip, X., 2018. A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. *ACM Computing Surveys,* 51(6).

Gupta, P., Pol, S., Rahatekar, D. & Patil, A., 2016. Smart Ambulance System. *International Journal of Computer Applications Proceedings on National Conference in Computing, Communication and Networking,* Juni, ACCNET 2016(6), pp. 23-26.

Kodali, R. K. & Mahesh, K. S., 2016. *A Low Cost Implementation of MQTT Using ESP8266.* Noida, IEEE.

Lee, S., 2014. Role of Parallelism in Ambulance Dispatching. *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* 44(8), pp. 1113-1122.

Limantara, R., Herjunianto & Roosalina, A., 2015. Fakto-faktor yang Mempengaruhi Tingginya Angka Kematian di UG Rumah Sakit. *Jurnal Kedokteran Brawijaya,* 28(2), pp. 200-205.

Mukhtar, M., 2015. GPS based Advanced Vehicle Tracking and Vehicle Control System. *International Journal of Intelligent Systems Technologies and Application,* 03(01), pp. 1-12.

Mulla, A., Baviskar, J., Baviskar, A. & Aniket, B., 2015. *GPS Assisted Standard Positioning Service for Navigation and Tracking: Review & Implementation.* Pune, IEEE.

Naik, N., 2017. *Choice of Effective Messaging Protocols for IoT: MQTT, CoAP, AMQP and HTTP.* Vienna, IEEE.

Oh, S., Kim, J.-H. & Fox, G. C., 2009. Real-Time Performance Analysis for Publish/Subscribe Systems. *Future Generation Computer Systems,* 26(3), pp. 318-323.

Peraturan Menteri Kesehatan Republik Indonesia Nomor 47 Tahun 2018 Pelayanan Kegawatdaruratan. 1 November 2018. Jakarta.

Salim, K. & Idrees, I. M., 2013. Design and Implementation of Web-Based GPS-GPRS Vehicle Tracking System. *IJCSET,* 3(12), pp. 443-448.

Samani, H. & Zhu, R., 2016. Robotic Automated External Defibrillator Ambulance for Emergency Medical Service in Smart Cities. *IEEE Access,* Volume 4, pp. 268-283.

Sarbpreet, Tripathy, S. & Mathew, J., 2016. *Design and evaluation of an IoT enabled secure multi-service Ambulance Tracking System.* Singapore, IEEE, pp. 2209-2214.

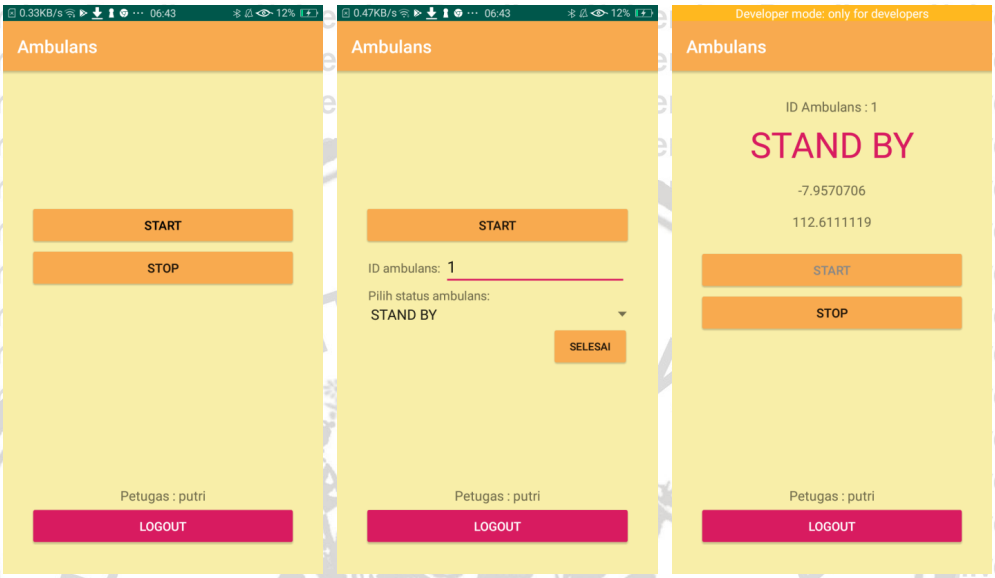
Saudin, D., Agoes, A. & Rini, I. S., 2016. Analisis Faktor yang Mempengaruhi Keterlambatan dalam Mengatasi Pasien Stroke Saat Merujuk ke RSUD Jombang. *Jurnal Hesti Wira Sakti,* pp. 1-12.

Tarigan, S. O. F., Sitepu, H. I. & Hutagalung, M., 2014. Pengukuran Kinerja Sistem Publish/Subscribe Menggunakan Protokol MQTT (Message Queuing Telemetry Transport). *Jurnal Telematika,* Volume 6, pp. 1858-2516.

LAMPIRAN A TAMPILAN APLIKASI

A.1 Aplikasi Node Publisher

Gambar A.1 merupakan tampilan aplikasi ketika melakukan publish data. Publish data dilakukan dengan memilih tombol start dan memasukan ID ambulans serta memilih status dari ambulans.

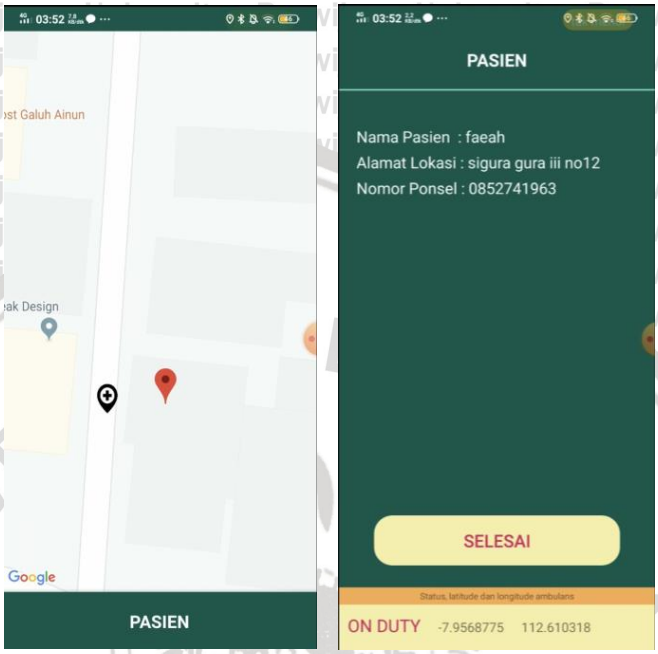


Gambar A. 1 Tampilan Aplikasi ketika Publish Data

Gambar A.2 adalah tampilan aplikasi ketika menerima pesanan dari pasien. Pengguna dapat memilih untuk menerima melakukan penjemputan pasien atau menolak penjemputan pasien.

Gambar A. 2 Tampilan Aplikasi Ketika Menerima Pesanan

Pada Gambar A.3 adalah tampilan ketika memilih menerima pesanan untuk menjemput pasien. Ketika menerima, maka aplikasi akan menampilkan maps. Pada maps tersebut terdapat penanda lokasi ambulans dan pasien. Serta dibawah maps terdapat tulisan "Pasien" yang ketika dipilih akan menampilkan data pasien dan tombol "Selesai". Tombol "Selesai" dipilih ketika pasien selesai di antarkan.



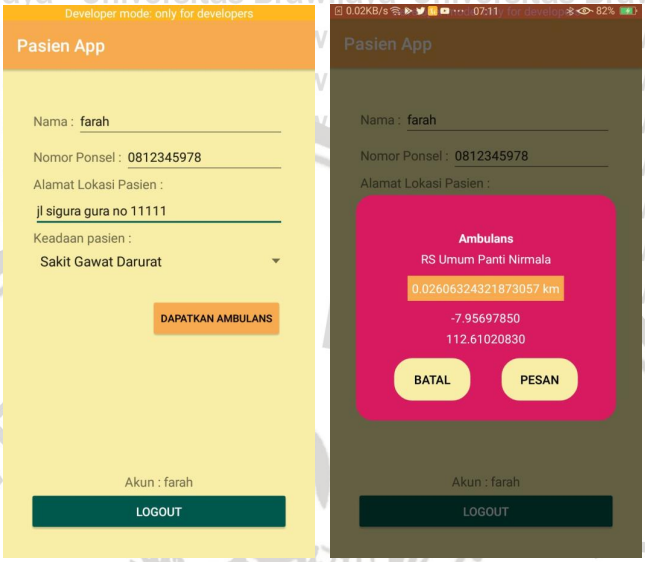
Gambar A. 3 Tampilan Aplikasi Ketika Menerima Pesanan

Pada gambar Gambar A.4 adalah tampilan ketika memilih menolak pesanan untuk menjemput pasien. Ketika menolak, maka aplikasi akan menampilkan pilihan alasan menolak.

Gambar A. 4 Tampilan Aplikasi Ketika Menolak Pesanan

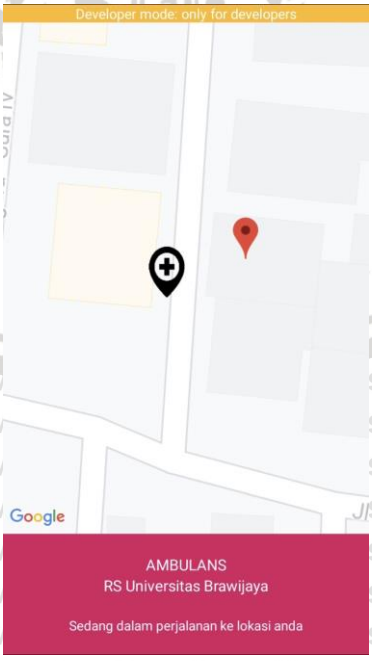
A.2 Aplikasi Perangkat Bergerak Pasien

Gambar A.5 adalah tampilan ketika pengguna ingin meminta memesan ambulans. Pengguna memasukan data diri kemudian memilih tombol "DAPATKAN AMBULANS". Setelah menekan tombol tersebut, akan muncul data ambulans. Pengguna dapat memilih tombol "BATAL" jika tidak ingin memesan ambulans dan tombol "PESAN" jika ingin memesan ambulans.



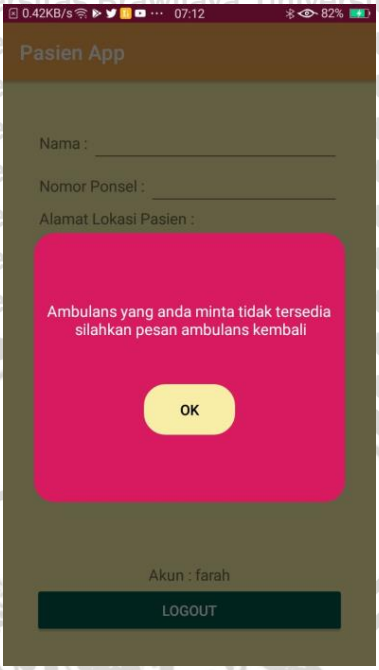
Gambar A. 5 Tampilan Aplikasi Ketika Memesan Ambulans

Gambar A.6 adalah tampilan aplikasi pengguna ketika pengguna memesan ambulans dan pesanan diterima. Aplikasi akan menampilkan *maps* dengan penanda titik ambulans dan titik pengguna.



Gambar A. 6 Tampilan Aplikasi Ketika Pesanan Diterima

Gambar A.7 adalah tampilan aplikasi ketika pengguna memesan ambulans dan pesanan ambulans ditolak.



Gambar A. 7 Tampilan Aplikasi Ketika Pesanan Ditolak

LAMPIRAN B *RESPONSE TIME* DARI NODE APLIKASI PASIEN KE NODE SUBSCRIBER (WEB SERVER)

Tabel B.1 adalah hasil pengujian *response time.* Pengujian *response time* ini menghitung waktu yang diperlukan dari aplikasi pasien ke node subscriber (web server) untuk menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans pada basis data adalah 1000 data.

Tabel B. 1 Hasil Pengujian Response Time 1000 Data

Percobaan	Response Time (milliseconds)
1	52
2	67
3	61
4	53
5	67
6	124
7	68
8	135
9	69
10	97
11	53
12	78
13	61
14	54
15	74
16	68
17	67
18	49
19	55
20	60
21	61
22	74
23	60
24	63
25	58
26	61
27	68
28	97
29	81
30	67
Rata-rata	70,06666667

Tabel B.2 adalah hasil pengujian *response time.* Pengujian *response time* ini menghitung waktu yang diperlukan dari aplikasi pasien ke node subscriber (web server) untuk menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans pada basis data adalah 2000 data.

Tabel B. 2 Hasil Pengujian Response Time 2000 Data

Percobaan	Response Time (milliseconds)
1	90
2	108
3	143
4	97
5	116
6	91
7	105
8	91
9	87
10	163
11	173
12	120
13	104
14	121
15	110
16	132
17	123
18	103
19	102
20	108
21	94
22	98
23	118
24	106
25	81
26	96
27	99
28	91
29	94
30	105
Rata-rata	109,6206897

Tabel B.3 adalah hasil pengujian *response time.* Pengujian *response time* ini menghitung waktu yang diperlukan dari aplikasi pasien ke node subscriber (web server) untuk menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans pada basis data adalah 3000 data.

Tabel B. 3 Hasil Pengujian Response Time 3000 Data

Percobaan	Response Time (milliseconds)
1	150
2	140
3	123
4	130
5	158
6	164
7	142
8	130
9	123
10	186
11	151
12	138
13	131
14	133
15	138
16	160
17	144
18	142
19	148
20	125
21	146
22	161
23	149
24	128
25	182
26	137
27	129
28	136
29	165
30	122
Rata-rata	143,7

Tabel B.4 adalah hasil pengujian *response time.* Pengujian *response time* ini menghitung waktu yang diperlukan dari aplikasi pasien ke node subscriber (web server) untuk menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans pada basis data adalah 4000 data.

Tabel B. 4 Hasil Pengujian Response Time 4000 Data

Percobaan	Response Time (milliseconds)
1	184
2	215
3	208
4	203
5	190
6	168
7	194
8	174
9	179
10	203
11	186
12	204
13	163
14	200
15	200
16	229
17	159
18	214
19	193
20	178
21	162
22	158
23	188
24	159
25	183
26	186
27	217
28	183
29	162
30	197
Rata-rata	187,9666667

Tabel B.5 adalah hasil pengujian *response time.* Pengujian *response time* ini menghitung waktu yang diperlukan dari aplikasi pasien ke node subscriber (web server) untuk menentukan ambulans yang dibutuhkan oleh pasien dengan variasi jumlah data ambulans pada basis data adalah 5000 data.

Tabel B. 5 Hasil Pengujian Response Time 5000 Data

Percobaan	Response Time (milliseconds)
1	226
2	239
3	284
4	246
5	212
6	266
7	221
8	248
9	240
10	247
11	256
12	253
13	255
14	247
15	243
16	227
17	233
18	214
19	235
20	216
21	248
22	225
23	229
24	221
25	235
26	228
27	239
28	227
29	217
30	269
Rata-rata	238,2