

BAB I

PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi sangat berkembang dengan pesat di kehidupan kita. Hal itu ditunjukkan dengan berbagai bukti seperti melihat banyaknya anak-anak yang terbiasa memegang sebuah *gadget*, lalu diikuti dengan munculnya berbagai alat canggih untuk mengakses *virtual reality* dan sebagainya. Teknologi yang canggih memberikan keleluasaan terhadap para pengguna (*user*) untuk melakukan banyak hal hanya dengan sentuhan layar pada *smartphone* atau tablet mereka dimanapun dan kapanpun. Adanya akses internet juga memudahkan mereka untuk melakukan transaksi dan komunikasi dengan berbagai kemudahan. Menurut laporan dari Emarketer, Indonesia akan menjadi pengguna *smartphone* terbesar keempat di dunia setelah China, India dan Amerika Serikat dengan angka perkiraan lebih 100 juta pengguna aktif pada tahun 2018. Laporan tersebut juga telah menegaskan android sebagai *platform* dominannya (Emarketer, 2014). Bukan hanya itu, rata-rata jumlah jam pemakaian internet di Indonesia mencapai 5 jam per hari nya (Statista, 2015). Hal ini tentu saja menakjubkan dan menjadi sorotan internasional. Pada seminar IDG Vietnam, Indonesia dianggap sebagai “*Asia’s Next Big opportunity*” karena tingginya angka dalam perkembangannya (*growth rate*) dan banyaknya pengguna (*user*) *gadget* dengan akses internet di usia rata-rata 25 tahun (Vu, 2003). Dalam seminarnya, IDG Venture Vietnam menargetkan Indonesia sebagai target pasar untuk bidang aplikasi mereka, karena android terbukti sebagai OS (*Operating System*) yang mendominasi populasi pengguna *smartphone* di Indonesia.

Ketua Umum APJII (Asosiasi Penyelenggara Jasa Internet Indonesia), Samuel A Pangerapan, mengatakan bahwa data yang tercatat untuk tahun 2014 lalu adalah 78,5% dari 88,1 juta pengguna internet di Indonesia tinggal di wilayah

Indonesia bagian barat, dan 65% pengguna sekaligus penetrasi paling tinggi berasal dari kota DKI jakarta (Maulana, 2015). Kecanggihan teknologi seharusnya juga dapat dimanfaatkan di berbagai bidang tidak terkecuali bidang kesehatan. Salah satu cara memanfaatkan teknologi yang dapat diterapkan pada bidang kesehatan adalah dalam proses pemesanan ambulans. Ambulans merupakan sebuah kendaraan khusus untuk mengangkut orang sakit atau terluka ke rumah sakit (Dictionary, 2002). Pasien dapat memesan ambulans secara manual dengan cara menelpon rumah sakit terdekat ataupun instansi kesehatan lainnya yang juga menyediakan ambulans gratis maupun berbayar. Dinkes (Dinas Kesehatan) DKI Jakarta telah menunjang dan meringankan kebutuhan warga DKI Jakarta dengan cara memberikan layanan gratis ambulans bagi warga dengan KTP (Kartu Tanda Penduduk) berdomisili di DKI Jakarta. Pemerintah Provinsi DKI Jakarta telah bekerjasama dengan unit AGD (Ambulans Gawat Darurat) 118, menjadi AGD Dinkes, melalui Peraturan Gubernur No.144 tahun 2010 untuk mewujudkan sarana ambulans gratis bagi warga DKI Jakarta (AGD Dinkes, 2013). Bila seseorang tidak memiliki KTP DKI Jakarta, maka warga cukup menyertakan Kartu Keluarga (KK) dan surat keterangan domisili dari pengurus setempat.

Aplikasi OAM (*Order Ambulance Online*) adalah aplikasi yang memudahkan warga DKI Jakarta dalam melakukan pemesanan ambulans menggunakan *gadget* berbasis android. OAM dapat diterapkan pada rumah sakit maupun instansi lain di bidang kesehatan dan ambulans. Melakukan pemesanan ambulans dengan OAM dapat meminimalisir kesalahan dalam proses *input data* secara manual mengenai lokasi dan data pribadi pasien (*user*) yang dapat saja terjadi dalam realisasinya dan memperlambat proses pemesanannya. Pemesanan ambulans dengan menuliskan informasi dasar pasien (*user*) pada aplikasi OAM di *gadget* berbasiskan android dapat memperkecil masalah *human error* yang dapat terjadi saat memesan ambulans melalui telepon. Fitur tersebut juga sekaligus dapat mempercepat proses penanganan pasien dengan menekan *response time* yang ada.

Waktu yang dibutuhkan dalam penanganan pasien, *response time*, juga sangat berpengaruh dalam tingkat keselamatan pasien. *Response time* pemesanan ambulans di Inggris terhadap telepon kategori A (yang mengancam nyawa) harus segera diatasi dalam waktu 8 menit. Sementara pasien dengan berkebutuhan khusus pada kendaraan harus diatasi dalam 19 menit (NHS England, 2014). India telah menanggapi serius mengenai *response time* pada ambulans mereka. Manish Sacheti, salah satu dari pendiri ZHL (Ziqitza Healthcare Ltd) yang bergerak di bidang ambulans dengan hotline 1298, berharap agar *response time* pada ambulans dapat di proses lebih cepat menjadi 7 atau 8 menit dari *response time* saat ini yaitu hanya 15 menit (DSilva, 2007).

Sejak tahun 2006 hingga saat ini, AGD 118 memiliki catatan *response time* 10 hingga 20 menit. Hal ini menunjukkan bahwa *response time* merupakan catatan yang penting dan sulit ditekan (Nugroho, 2016). Tingkat angka kematian akan semakin tinggi jika *response time* yang dibutuhkan instansi penyedia ambulans semakin besar. *Response time* pada proses pemesanan ambulans dapat dipersingkat dengan cara melakukan efektifitas pada saat *input* data dasar pasien (*basic information*) melalui OAM dibandingkan melalui telepon, mengetahui lokasi secara langsung menggunakan GPS pada aplikasi OAM dibandingkan pemberian nama jalan kepada *driver* / pengemudi ambulans, dan sortir tingkat kebutuhan untuk memprioritaskan pasien dalam keadaan darurat. Validasi lokasi melalui metode GIS (*Geographic Information System*) dan GPS (*Global Positioning System*) dapat mempermudah pasien (*user*) dan admin ambulans dalam prosesnya. Pengemudi (*driver*) ambulans akan memiliki gambaran lebih jelas mengenai lokasi pasien. Banyak hal yang dapat mempengaruhi perjalanan pengemudi ambulans, misalnya adalah *driver* merupakan seseorang yang baru di jakarta atau belum pernah mengemudi di daerah tersebut, hal tersebut akan memperlambat *response time*.

Untuk mengoptimalkan aplikasi, algoritma Dijkstra akan diterapkan pada OAM untuk membantu menentukan urutan pos terdekat dari lokasi pemesanan oleh pasien (*user*). Kelebihan algoritma Dijkstra adalah algoritma ini dapat menyelesaikan pencarian jarak terpendek antara dua buah simpul atau *a pair shortest path*, antara semua pasangan simpul atau *all pair shortest path*, simpul tertentu ke semua simpul yang lain atau *single source shortest path*, dan antara dua buah simpul melalui beberapa simpul atau *intermediate shortest path* (Ardiani, 2011). Dalam penelitian perbandingan algoritma Greedy dan Dijkstra, jarak yang diperoleh oleh algoritma Greedy di dalam penelitiannya adalah 27. Angka tersebut lebih besar bila dibandingkan dengan jarak yang di peroleh oleh algoritma Dijkstra yaitu 12. Bila dibandingkan, algoritma Greedy dan Dijkstra berdasarkan jarak lintasannya, algoritma *Dijkstra* menghasilkan jarak yang lebih kecil (Lubis, 2009). Algoritma Dijkstra juga digunakan dalam jurnal berjudul Perencaaan Rute Perjalanan di Jawa Timur dengan Dukungan GIS menggunakan Metode Dijkstra, karena hasil yang dikeluarkan oleh algoritma tersebut hanya satu rute saja yang merupakan rute terpendek dari satu kota ke kota lain di dalam penelitian tersebut (Gunandi, 2002).

False berarti tidak sesuai dengan kebenaran atau fakta (Reverso Dictionary, 2000). *False emergency call* adalah situasi dimana seseorang yang menghubungi suatu instansi dalam keadaan darurat tidak sesuai dengan kebenaran atau fakta. Dalam wawancara dengan bapak Wahyu, beliau menyebutkan bahwa mereka kerap sekali mendapatkan *false call* di AGD118. Meskipun angka telepon yang jahil tersebut tidak mencapai 50%, namun hal tersebut cukup mengganggu dan dapat menghambat bagi seseorang yang sedang dalam keadaan benar-benar membutuhkan ambulans. Aplikasi OAM diharapkan dapat meminimalisir hal ini dengan cara adanya pengisian form pendaftaran aplikasi saat melakukan pemesanan pertama kali. Cara lain yang dapat dilakukan untuk menghindari hal ini adalah dengan menerapkan *user agreement* terhadap aplikasi OAM, sehingga mereka dapat dilegalkan untuk dikenakan sanksi.

1.2 Rumusan Masalah

Dalam penelitian ini masalah yang dibahas adalah sebagai berikut:

1. Bagaimana proses mendapatkan informasi dan lokasi untuk ambulans dan pasien (*user*) di Kota DKI Jakarta, sehingga dapat diolah menjadi satu aplikasi pemesanan ambulans berbasis *mobile*?
2. Bagaimana aplikasi ini dapat membantu memecahkan masalah yang dihadapi *admin* ambulans Jakarta dalam menentukan pos ambulans terdekat dari pasien (*user*) menggunakan algoritma Dijkstra?

1.3 Batasan Masalah

Batasan dari penelitian ini adalah sebagai berikut:

1. Aplikasi menggunakan O.S Android versi 4.4.2, Kitkat.
2. Implementasi aplikasi hanya mencakup di wilayah DKI Jakarta.
3. Server menggunakan *hosting* dan domain dari Digital Ocean dan Domainesia.
4. Menggunakan GPS bawaan dari smartphone.
5. Keamanan data tidak terlalu diperhatikan aplikasi tidak membahas tentang *tracking* pasien (*user*).
6. Jaringan dan security pada *device* tidak terlalu diperhatikan
7. Penentuan ketersediaan ambulans tiap pos ambulans dilakukan secara manual

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

1. Menghasilkan perancangan aplikasi OAM secara konseptual dan realisasinya untuk membantu masyarakat Kota DKI Jakarta sebagai pasien (*user*) untuk dapat melakukan pemesanan ambulans.
2. Membangun aplikasi OAM yang dapat digunakan untuk membantu *admin* ambulans DKI Jakarta dalam memecahkan masalah menentukan posisi ambulans terdekat dari pasien (*user*) dari ambulans – ambulans yang tersedia menggunakan algoritma Dijkstra.

1.5 Manfaat Penelitian

Manfaat dari penelitian yang penulis lakukan adalah sebagai berikut:

1. Memberikan hasil sebuah aplikasi OAM berbasis *mobile* yang dapat membantu masyarakat DKI Jakarta dalam melakukan pemesanan ambulans di wilayah DKI Jakarta.
2. Memberikan hasil sebuah aplikasi OAM berbasis *mobile* yang dapat membantu *admin* ambulans DKI Jakarta dalam menentukan posisi ambulans terdekat dari pasien (*user*) menggunakan algoritma Dijkstra

BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

Penelitian ini didasari oleh pemikiran dan rujukan pada penelitian – penelitian sebelumnya yang juga berkaitan dengan aplikasi pemesanan ambulans menggunakan metode GIS dan algoritma Dijkstra. Beberapa penelitian sebelumnya adalah :

1. Penelitian yang dilakukan oleh Farly Nur Dewantara dengan judul penelitian *Perancangan Aplikasi On-Call Positioning Menggunakan GIS (Geographic Information System) dan GPS pada Dinas Pemadam Kebakaran Bandung*. Penelitian ini menghasilkan aplikasi mobile pelaporan kebakaran kepada petugas pemadam kebakaran di wilayah Bandung. Karena ada banyaknya penelpon yang tidak bertanggung jawab terjadi pada Damkar Bandung, maka dibutuhkan aplikasi yang dapat membantu memvalidasi kebenaran laporan pemadam kebakaran. Aplikasi juga disertai *web based app* untuk admin pemadam kebakaran. Aplikasi ini hanya menggunakan metode GIS. Sistem aplikasi dibangun dengan bahasa pemrograman Android, PHP, dengan sistem *database MySQL*. (Dewantara, 2013)
2. Penelitian yang dilakukan oleh Imron Fauzi dengan judul *Penggunaan Algoritma Dijkstra Dalam Pencarian Rute Tercepat dan Rute Terpendek (Studi kasus: Pada Jalan Raya antara Wilayah Blok M dan Kota)*. Penelitian ini menghasilkan aplikasi desktop untuk menentukan rute tercepat dan terpendek, dalam wilayah kota Jakarta. Wilayah aplikasi diperuntukkan pada jalan raya antara wilayah Blok M dan Kota. Penelitian ini juga membandingkan algoritma Dijkstra, Bellman Ford, dan Floyd Warshall. Setelah melakukan perbandingan – perbandingan terhadap beberapa algoritma, penulis

menyimpulkan untuk menggunakan algoritma Dijkstra karena kompleksitas waktu yang lebih kecil. Algoritma Dijkstra juga tidak memiliki bobot negatif. Aplikasi di bangun menggunakan bahasa pemrograman PHP dan *database* mySQL. (Fauzi, 2011)

3. Penelitian yang dilakukan oleh Stevian Suryo Saputro dengan judul *Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata di Kota Manado berbasis Moble Web Dengan Algoritma Dijkstra*. Penelitian ini menghasilkan *mobile* web yang dapat menampilkan rute antar satu tempat wisata dengan tempat wisata lainnya, serta rute dari posisi *user* menuju posisi hotel-hotel di kota Manado. Aplikasi yang di bangun menggunakan metode GIS dan algoritma Dijkstra (Saputro, 2013).

Tabel 2.1 Penelitian Terdahulu dan Penelitian Penulis

Judul	Pengarang	Keterangan
Perancangan Aplikasi <i>On-Call Positioning</i> Menggunakan GIS (<i>Geographic Informarion System</i>) dan GPS pada Dinas Pemadam Kebakaran Bandung	Farly Nur Dewantara (Dewantara, 2013)	Aplikasi pelaporan pemadam kebakaran di wilayah bandung berbasis android menggunakan GIS.
Penggunaan Algoritma Dijkstra Dalam Pencarian Rute Tercepat dan Rute Terpendek (Studi kasus: Pada Jalan Raya antara Wilayah Blok M dan Kota)	Imron Fauzi (Fauzi, 2011)	Aplikasi berbasis desktop untuk pencarian jalur tercepat pada jalan raya antara wilayah Jakarta. Penelitian dibatasi hanya pada wilayah Blok M dan Kota.

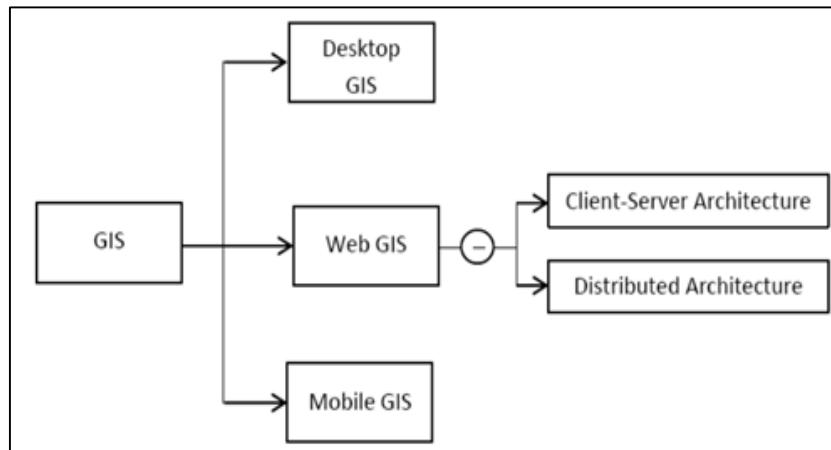
Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata di Kota Manado berbasis <i>Mobile Web</i> Dengan Algoritma Dijkstra.	Stevian Suryo Saputro (Saputro, 2013)	Aplikasi berbasis mobile web yang dapat menampilkan rute antar satu tempat wisata dengan tempat wisata lainnya, serta rute dari posisi user menuju posisi hotel di kota manado.
--	--	---

2.2 GIS (*Geographic Information System*)

GIS atau *Geographic Information System* (dalam bahasa Indonesia disebut Sistem Informasi Geografis) merupakan sistem informasi yang mengelola data yang memiliki informasi spasial. GIS merupakan sistem komputer yang memiliki kemampuan khusus untuk membangun, menyimpan, mengelola dan menampilkan informasi berasal dari geografi (Riyanto, 2010). Sistem ini berkaitan erat dengan informasi geografis sehingga ada juga yang mendefinisikan GIS sebagai sistem informasi yang dapat memberikan informasi lokasi dan waktu mengenai suatu kejadian atau aktivitas (Longley, 2011).

Berdasarkan teknologi dan implementasinya, GIS (*geographic information system*) dapat dikategorikan dalam 3 aplikasi yaitu GIS berbasis *desktop* (*Desktop GIS*), GIS berbasis *web* (*Web GIS*), dan GIS berbasis *mobile* (*Mobile GIS*). Lihat gambar 2.1. GIS berbasis *desktop* (*Desktop GIS*) memiliki keterbatasan penggunaan yang hanya terbatas untuk komputer *desktop*. Tidak semua pengguna dapat mengaksesnya karena ini merupakan aplikasi yang berdiri sendiri (*stand alone*). GIS berbasis *desktop* (*Desktop GIS*) juga memiliki beberapa kemampuan yaitu kemampuan untuk menampilkan data peta, analisis data, dan mempublikasi (*publish*) data. GIS berbasis *web* (*Web GIS*) merupakan sistem informasi geografis yang didistribusikan pada seluruh lingkungan jaringan komputer untuk melakukan integrasi, penyebaran, dan melakukan komunikasi informasi geografis secara *visual* di *World Wide Web*.

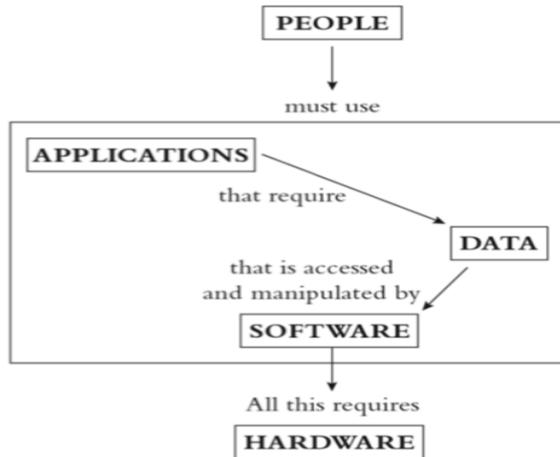
(WWW) melalui internet. GIS berbasis *mobile* (*Mobile GIS*) merupakan sistem informasi geografis yang diimplementasikan pada perangkat *mobile* dengan keterbatasan ruang penyimpanan dan tampilan. Implementasi GIS berbasis *mobile* (*Mobile GIS*) dapat dilakukan sebagai aplikasi yang berdiri sendiri (*stand alone*) dengan penyimpanan data dalam perangkat *mobile* atau implementasi juga dapat dilakukan dengan menyesuaikan arsitektur *servernya* (GIS berbasis *web* / *Web GIS*). Ketiga kategori GIS, yaitu GIS berbasis *desktop* (*Desktop GIS*), GIS berbasis *web* (*Web GIS*), dan GIS berbasis *mobile* (*Mobile GIS*), tetap memiliki hubungan satu dengan lainnya meskipun telah dikategorikan menjadi tiga bagian (Riyanto, 2010).



**Gambar 2.1 Kategori *Geographic Information System*
(Riyanto, 2010)**

2.2.1 Komponen GIS

Komponen-komponen GIS terdiri atas manusia (*people*), aplikasi (*applications*), informasi data (*data informations*), perangkat lunak (*software*), dan perangkat keras (*hardware*). Gambar 2.2 menggambarkan komponen-komponen GIS (Harmon, 2003). Deskripsi mengenai komponen – komponen GIS adalah sebagai berikut :



**Gambar 2.2 Komponen-komponen GIS
(Harmon, 2003)**

A. Manusia (*People*)

Manusia (*People*) adalah pengguna sistem dan merupakan komponen yang paling penting. Karena proses desain dan implementasi sistem GIS dimulai oleh manusia dan kebutuhannya. Manusia juga berfungsi untuk membuat standar, membuat *update data*, membuat aplikasi, dan analisa *output* untuk hasil yang diinginkan.

B. Aplikasi (*Applications*)

Aplikasi (*Applications*) adalah proses dan program yang digunakan untuk melakukan suatu pekerjaan. Aplikasi termasuk menjadi komponen kedua karena mereka menetapkan langkah-langkah yang harus diselesaikan dalam pekerjaan.

C. Informasi Data (*Data Informations*)

Informasi Data (*Data Informations*) merupakan informasi berisikan data – data yang dibutuhkan untuk menunjang aplikasi. Data yang ditangani dalam GIS

merupakan data spasial, yaitu sebuah data yang berorientasi geografis. Tipe data spasial memiliki 2 bagian yang penting, yaitu infomasi lokasi (spasial) dan informasi deskriptif (atribut).

1. Informasi lokasi (spasial) merepresentasikan obyek di bumi menggunakan referensi geografis baik koordinat geografi (lintang dan bujur) dan koordinat XYZ (Oktavia, t.th). Bagian ini memiliki beberapa bentuk data, yaitu:

Tabel 2.2 Bentuk-bentuk data informasi lokasi (spasial)
(Oktavia, t.th)

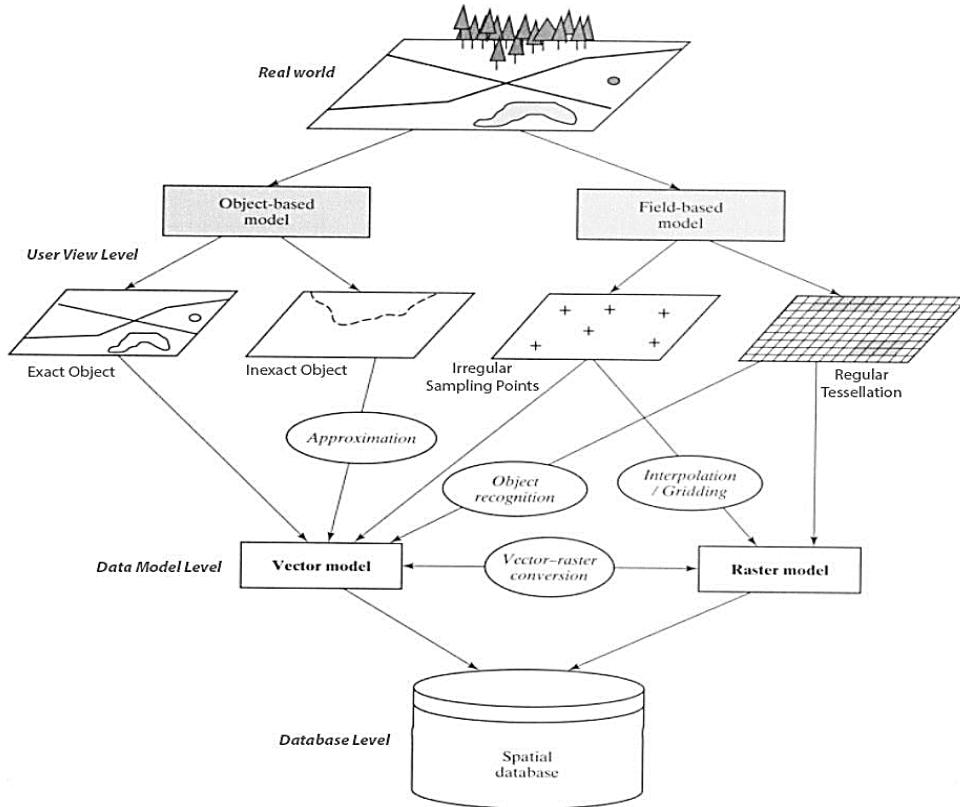
Bentuk-bentuk		
informasi lokasi	Deskripsi	Format Titik
Titik (<i>Dot</i>)	Bentuk data ini merupakan bentuk data paling sederhana bagi obyek spasial. Tipe data ini tidak memiliki dimensi. Contoh : Lokasi kecelakaan	Format titik : Koordinat tunggal. Tanpa panjang/jarak. Tanpa luas
Garis (<i>Polyline</i>)	Garis merupakan bentuk geometri linier yang akan menghubungkan dua atau lebih titik. Tipe data ini merepresentasikan obyek dalam satu dimensi. Contoh : Jalan, Sungai.	Format titik : Ada koordinat titik awal dan akhir. Ada panjang/jarak. Tanpa luas.
Area (<i>Polygon</i>)	Tipe data ini merepresentasikan obyek dalam dua dimensi. Contoh : Luas tanah.	Format titik : Koordinat dengan titik awal dan akhir sama. Ada panjang/jarak. Ada luas.

Permukaan (3D)	Tipe data ini merepresentasikan obyek dalam tiga dimensi. Contoh : Peta slope, bangunan bertingkat	Format titik : Area dengan koordinat vertikal, Area dengan ketinggian.
---------------------------	--	--

2. Informasi deskriptif (atribut) merupakan data yang merepresentasikan deskripsi atau penjelasan yang berkaitan dari suatu lokasi. Bagian ini memiliki beberapa bentuk data, yaitu :

- a) Format Tabel memiliki bentuk data berupa kata-kata, kode alfanumerik, angka. Contoh : Hasil proses, Indikasi, Atribut.
- b) Format Laporan memiliki bentuk data berupa teks, deskripsi. Contoh : Perencanaan, Laporan Proyek, Pembahasan.
- c) Format Pengukuran memiliki bentuk data berupa angka-angka, hasil. Contoh : Jarak, Inventarisasi, Luas.
- d) Format grafik anotasi memiliki bentuk data berupa kata-kata, symbol. Contoh: Nama objek, legend, grafik, peta.

Sehingga apabila ada suatu aktivitas untuk memperoleh data obyek pemukiman di suatu daerah, bentuk data spasialnya merupakan data grafik berbentuk poligon yang menghubungkan posisi-posisi geografis. Sedangkan data atributnya mendapatkan informasi data berupa luas permukiman, jumlah penduduk, jumlah rumah dll. Perhatikan gambar 2.3 untuk gambaran pembagian jenis-jenis data dari dunia nyata (*real world*) hingga *database* nya. Dalam data spasial, data dapat direpresentasikan ke dalam dua model yaitu:



**Gambar 2.3 Vektor dan Raster data model
(Lo, 2007)**

1. Data Vektor

Data vektor merupakan data yang menampilkan bentuk bumi yang direpresentasikan ke dalam kumpulan garis, area, titik dan nodes. Baik digunakan untuk analisa dengan ketepatan posisi yang tinggi. Kelemahan data ini adalah data vektor tidak mampu mengatasi perubahan gradual.

2. Data Raster

Data raster merupakan data yang merepresentasikan obyek geografis sebagai struktur sel grid yang disebut *pixel* (*picture element*). Baik digunakan untuk menggambarkan

batas-batas yang berubah secara gradual, misalnya tanah, kelembaban tanah, suhu, dan sebagainya. Kelemahan data ini adalah keterbatasan data akibat besarnya ukuran *file* yang dapat mempengaruhi kualitas resolusinya. Data raster juga bergantung pada kemampuan *hardware* dalam penggerjaannya.

Data spasial dapat didapatkan di beberapa sumber, diantaranya adalah peta analog, data sistem penginderaan jauh, data hasil pengukuran lapangan, dan data GPS (*Global Positioning System*). Fungsi analisis spasial dapat memberikan informasi spesifik mengenai aktifitas ataupun peristiwa yang terjadi pada suatu daerah tertentu pada periode tertentu (Prahasta, 2014)

D. Perangkat lunak (*software*)

Perangkat lunak (*software*) merupakan inti dari *software GIS*, karena software menghasilkan fungsi dan alat yang dibutuhkan untuk menciptakan, menganalisis dan menampilkan informasi geografis. Contoh software GIS adalah *tools* untuk memasukkan dan memanipulasi data, sistem untuk mengelola basis data (*Database Management System / DBMS*), *tools* yang dapat mendukung query, analisis dan visualisasi geografis.

E. Perangkat keras (*hardware*)

Perangkat keras (*hardware*) merupakan komponen fisik yang menjadi tempat menjalankan sistem. Perangkat keras seperti komputer atau laptop berguna untuk menyimpan data dan melakukan pemrosesan data dalam GIS yang mendukung kebutuhan analisis spasial dan pemetaan.

2.2.2 Geocoding dan Geocoder

Geocoding merupakan proses pengambilan data koordinat geografis berupa titik lintang (*latitude*) dan titik bujur (*longitude*) (Boscoe, 2007). *Geocoder* merupakan layanan yang menerima *address query* / alamat untuk menjalankan tugas-tugas dari proses *geocoding*, kemudian menghasilkan sebuah *output* dalam bentuk titik koordinat (Teske, 2014). *Geocoder* dibutuhkan untuk mendukung proses *geocoding*. Beberapa *online geocoder* yang menyediakan layanan gratis dan berbayar adalah Google, Bing dan Yahoo!. Layanan-layanan tersebut menyediakan *geocoding web service*. Kualitas tiap-tiap *geocoder* tidak memiliki standar penilaian yang universal, sehingga dalam perbandingannya tidak dapat secara langsung dikonklusikan (Goldberg, 2011). Beberapa faktor yang dapat membedakan fitur dari Google, Bing, dan Yahoo! terdapat di tabel 2.3.

Tabel 2.3 Perbandingan *Online Geocoding API*
(Xu, 2012)

	Google Geocode Service	Bing Geocode Service	Yahoo! Geocode Service
Contoh API request	http://maps.googleapis.com/maps/api/geocode/xml?address=500+108th+ave+ne,+bellevue+w+98004&sensor=true	http://dev.virtualearth.net/REST/v1/Location/US/WA/98004/bellevue/500+108th+ave+n?key=[APIkey]	http://yboss.yahooapis.com/geo/placefinder?location=701+First+Ave,+Sunnyvale,+CA
Sumber data (Data resources)	Internal Street Networks	TeleAtlas, Navteq, Map Data Sciences	Navteq

	Google Geocode Service	Bing Geocode Service	Yahoo! Geocode Service
Kapasitas request (Request capacity)	2500 <i>request</i> per hari	<i>Trial</i> - 10000 <i>request</i> per bulan <i>Basic</i> - 125000 <i>request</i> per tahun <i>Enterprise</i> - lebih dari 125000 <i>request</i> per tahun	Dapat mencapai 35000 (dan lebih dari itu) <i>request</i> per hari, tergantung pada biaya yang ditawarkan
Presisi geocoding (Geocoding precision)	Rooftop, Geometric_Center, Range_Interpolated, Approximate, Zero_Results	Parcel, Interpolation, Rooftop, InterpolationOffset, Null	99, 90, 87, 86, 85, 84, 82, 80, 75, 74, 72, 71, 70, 64, 63, 62, 60, 59, 50, 49, 40, 39, 30, 29, 20, 19, 10, 9 , 0
Return Format	JSON/XML	JSON/XML	XML

Setelah melihat beberapa faktor yang dapat membedakan fitur dari Google, Bing, dan Yahoo! terdapat di tabel 2.3, dapat terlihat bahwa pengambilan data seperti garis bujur dan garis lintang dari nama lokasi dalam penelitian pembuatan aplikasi OAM ini akan menggunakan Google *geocoding*, karena *geocoder* ini memiliki beberapa keunggulan. Google API (*Application Programming Interface*) tidak mengharuskan melakukan pemisahan data dalam komponen tertentu, seperti jalan dan kota, karena sudah ada proses *parsing* didalamnya. Google *geocoding* juga memiliki akurasi yang bagus (Sturgeon, 2011). Para peneliti di RTI International, North Carolina, telah menggunakan Google API untuk melakukan banyak bentuk kalkulasi

GIS (*geographic information system*) pada penelitian mereka (Trahan, 2009). Google *geocoding* juga memiliki keunggulan lain, karena di dalam *geocoder* ini juga sudah terdapat proses normalisasi dan standardisasi huruf kapital, tanda baca, dan singkatan / kependekan (Majewski, 2016).

2.3 Android

Android adalah sebuah sistem operasi untuk perangkat mobile berbasis linux yang mencakup sistem operasi, *middleware* dan aplikasi. Android menyediakan *platform* terbuka bagi para *developer*. Beberapa keunggulan android adalah android merupakan *platform mobile* pertama yang lengkap, Terbuka, dan Bebas (Safaat, 2012).

1. Lengkap (*Complete Platform*)

Para programmer dan *developer* dapat melakukan pengembangan aplikasi dengan leluasa ketika mereka sedang menggunakan platform android, karena bentuk *platform* ini merupakan bentuk yang lengkap yang mereka dapatkan.

2. Terbuka (*Open Source Platform*)

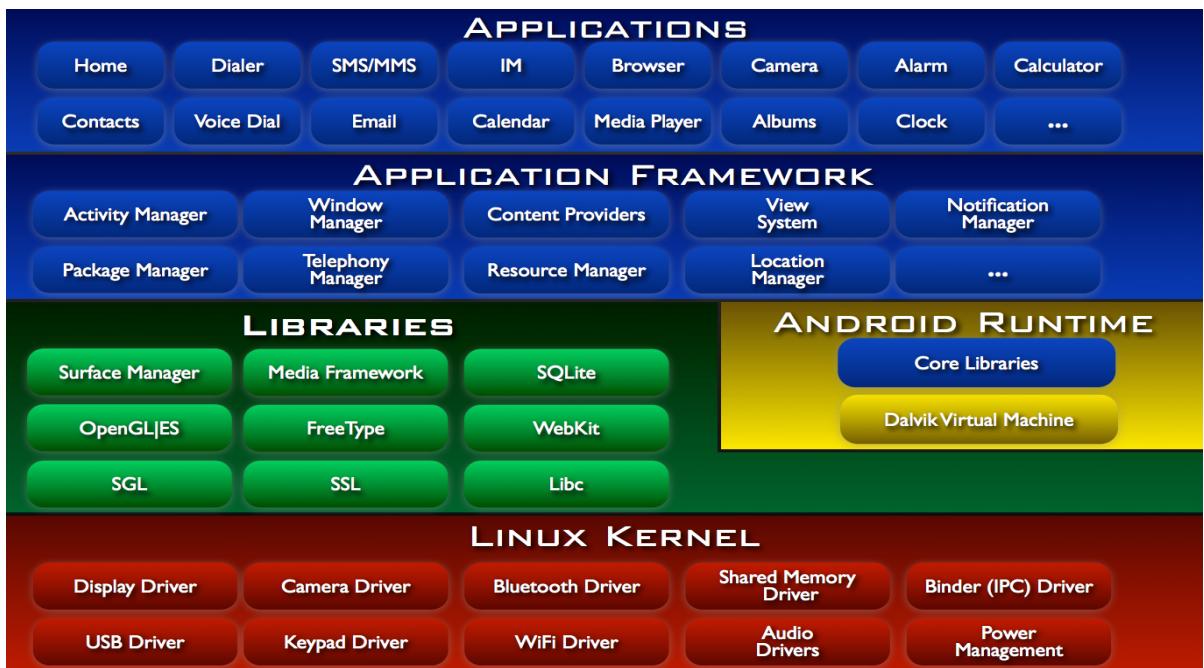
Platform android disediakan melalui lisensi *open source*. Hal itu berarti, pengembang dapat dengan bebas untuk mengembangkan aplikasi. Aplikasi dengan lisensi *open source* biasanya lebih stabil, tidak memakan biaya yang banyak / tanpa biaya, memiliki komunitas (*support community*) yang besar sehingga dapat bertanya langsung dengan para ahli, dan memiliki *system requirement* yang rendah (Bernstein, 2004)

3. Free (*Free Platform*)

Android adalah platform/aplikasi yang bebas untuk di develop, tidak ada biaya royalty untuk dikembangkan pada platform android karena menggunakan lisensi *open source*.

2.3.1 Anatomi Android

Dalam paket sistem operasi Android terdiri atas beberapa bagian yang merupakan anatomi android, perhatikan gambar 2.4.



Gambar 2.4 Anatomi Android

(Google Code)

1. Linux Kernel

Meskipun android diciptakan menggunakan Linux kernel, namun android bukanlah linux. Hal itu berarti, android tidak memenuhi seluruh standar pada linux *utilities* karena android bukanlah linux. Linux kernel digunakan untuk menciptakan android karena linux kernel memiliki memori dan manajemen proses yang baik. Linux mendukung *shared libraries* dan merupakan lisensi *opensource*. Kernel linux menyediakan driver layar, kamera, keypad, WiFi, *Flash Memory*, audio, dan IPC (*Interprocess Communication*) untuk mengatur aplikasi dan lubang keamanan .

2. Native Libraries

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk ter tanam pada kernel Linux. Beberapa pustaka diantaranya adalah :

- a) *Media Library* untuk memutar dan merekam berbagai macam format audio dan video.
- b) *Surface Manager* untuk mengatur hak akses layer dari berbagai aplikasi.
- c) *Graphic Library* termasuk didalamnya SGL dan OpenGL, untuk tampilan 2D dan 3D.
- d) *SQLite* untuk mengatur relasi *database* yang digunakan pada aplikasi. Merupakan *back end* pada mayoritas penggunaan platform penyimpanan data.
- e) *SSL* dan *WebKit* untuk *browser* dan keamanan internet.

3. Android Runtime

Android Runtime merupakan mesin virtual yang membuat aplikasi Android menjadi lebih tangguh dengan paket pustaka yang telah ada. Dalam *Android Runtime* terdapat 2 bagian utama, diantaranya :

- a) Pustaka Inti (*Core Libraries*)
Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.
- b) Mesin Virtual Dalvik (*Dalvik Virtual Machine*)
Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi *file* dalam format Dalvik Executable (*.dex).

Dalvik *virtual machine* dapat mendukung banyak pemprosesan pada satu *device* karena menggunakan *runtime memory* secara sangat efisien.

4. *Application Framework*

Application framework menyediakan kelas-kelas yang dapat digunakan untuk mengembangkan aplikasi Android. Beberapa bagian terpenting dalam kerangka aplikasi Android adalah sebagai berikut :

a. *Activity Manager*

Berfungsi untuk mengontrol siklus hidup aplikasi dan menjaga keadaan *BackStack* untuk navigasi penggunaan.

b. *Content Providers*

Berfungsi untuk merangkum data yang memungkinkan digunakan oleh aplikasi lainnya, seperti daftar nama.

c. *Resource Manager*

Berfungsi untuk mengatur sumber daya yang ada dalam program. Serta menyediakan akses sumber daya di luar kode program, seperti karakter, grafik, dan *file layout*.

d. *Location Manager*

Berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.

e. *Notification Manager*

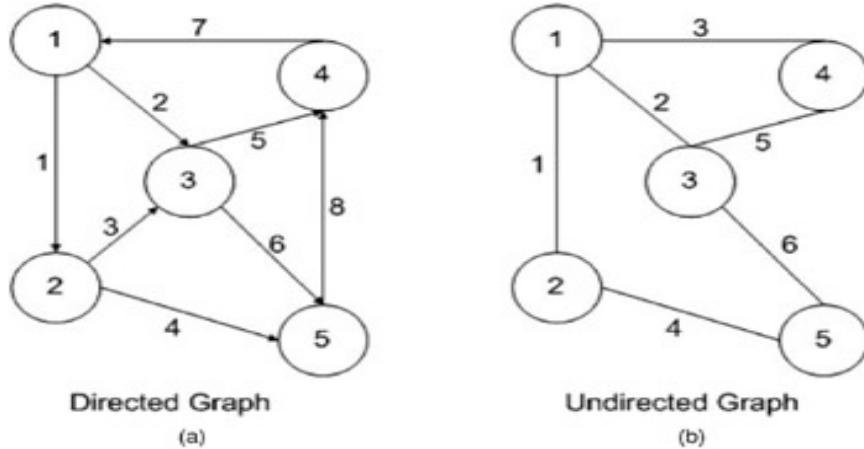
Hal ini mencakup berbagai macam peringatan (*reminder*) seperti, pesan masuk, janji, dan lain sebagainya yang akan ditampilkan pada *status bar*.

5. Application Layer

Barisan paling atas dari diagram arsitektur Android adalah lapisan aplikasi dan *widget*. Lapisan aplikasi merupakan lapisan yang terlihat pada pengguna ketika menjalankan program. Pengguna hanya akan melihat program ketika menggunakannya, tanpa mengetahui proses yang terjadi dibalik lapisan aplikasi. Lapisan ini berjalan dalam Android *runtime* dengan menggunakan kelas dan *service* yang tersedia pada *framework* aplikasi.

2.4 Algoritma Pencarian Jarak Terpendek (*Shortest Path Algorithm*)

Algoritma pencarian jarak terpendek (*shortest path algorithm*) merupakan solusi untuk mencari jarak terpendek pada lintasan atau rute dari titik awal hingga titik tujuan. Biasanya, untuk memudahkan penggambaran ilustrasinya digunakan graf (Magzhan, 2013). Sebuah graf merupakan objek matematika yang bersifat abstrak. Graf didefinisikan sebagai pasangan himpunan (V,E) , ditulis dengan notasi $G = (V,E)$. Dalam hal ini, V (*vertices* atau *nodes*) merupakan himpunan yang tidak kosong dari simpul-simpul. Lalu E (*edge* atau *arcs*) merupakan himpunan sisi yang menghubungkan sepasang simpul. Simpul (*vertex*) dapat dinyatakan sebagai huruf, bilangan asli $1, 2, 3, \dots$ dst, atau gabungan keduanya (Munir, 2005). Sedangkan sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan (u, v) . Jika e merupakan bagian sisi yang menghubungkan pasangan (u, v) , maka $e = (u, v)$. Secara umum klasifikasi pada graf berdasarkan orientasi arah pada sisinya dibagi atas graf ber arah (*directed graph*) dan graf tidak ber arah (*undirected graph*) (Fauzi, 2011). Gambar 2.5 (a) menunjukkan gambar graf berarah karena memiliki urutan lintasan yang harus diperhatikan. Gambar 2.5 (b) menunjukkan gambar graf yang tidak terarah, sehingga urutannya tidak diperhatikan.



Gambar 2.5 Klasifikasi graf berdasarkan orientasinya;

(a) Graf terarah / *Directed graph*

(b) Graf tidak terarah / *Undirected graph*

(Somani, 2005)

Dalam dunia nyata, sangat memungkinkan untuk mengaplikasikan teori graf dalam kehidupan sehari-hari. Misalnya, Graf dapat merepresentasikan peta dalam dunia nyata dalam menampilkan titik kota dan rute / jalur yang menghubungkan antar kota. Bila kita memiliki tujuan dan tempat asal, maka peta dapat diklasifikasikan menjadi graf terarah, dan sebaliknya. Ada beberapa algoritma yang popular dalam menyelesaikan masalah jarak terpendek yaitu algoritma Dijkstra, algoritma Bellman, dan algoritma Floyd (Magzhan, 2013).

2.4.1 Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger Dijkstra. Algoritma ini merupakan algoritma yang paling sering digunakan dalam pencarian rute terpendek. Penggunaannya dengan menggunakan simpul simpul sederhana pada jaringan jalan yang tidak rumit menjadikannya algoritma yang sederhana untuk digunakan (Chamero, 2006). Algoritma dijkstra merupakan salah satu varian dari algoritma Greedy, sehingga memiliki beberapa elemen penyusun algoritma Greedy (Novandi, 2007) yaitu :

a. Himpunan Kandidat C

Himpunan ini berisi elemen-elemen yang memiliki peluang atau kemungkinan untuk membentuk solusi. Pada persoalan pencarian lintasan terpendek (*shortest path*) dalam graf, himpunan simpul pada graf tersebut merupakan himpunan kandidat.

b. Himpunan Solusi S

Himpunan ini berisi solusi dari permasalahan yang diselesaikan. Himpunan solusi ini adalah bagian dari himpunan kandidat.

c. Fungsi Seleksi

Fungsi seleksi merupakan fungsi yang akan memilih tiap-tiap kandidat yang yang memungkinkan untuk menghasilkan solusi optimal pada setiap langkahnya.

d. Fungsi Kelayakan

Fungsi kelayakan merupakan fungsi yang akan memeriksa apakah suatu kandidat terpilih telah melanggar *constraint* atau tidak. Apabila kandidat melanggar *constraint* maka kandidat tidak akan dimasukkan ke dalam himpunan solusi.

e. Fungsi Objektif

Fungsi objektif merupakan fungsi yang akan memaksimalkan atau meminimalkan nilai solusi. Memilih hanya satu solusi terbaik dari masing - masing anggota himpunan solusi merupakan tujuan dari fungsi ini.

Kelemahan algoritma ini adalah algoritma ini tidak mampu mencari ujung / titik yang bernilai negatif (Sanan, 2013). Contoh implementasi algoritma ini adalah ketika G merupakan graf berarah dengan titik-titik $V(G) = \{v_1, v_2, \dots, v_n\}$. Algoritma Dijkstra akan mencari satu titik yang jumlah bobotnya paling kecil pada titik awal / titik 1. Titik-titik yang terpilih dipisahkan dan titik tersebut tidak diperhatikan lagi dalam langkah iterasi selanjutnya (Tiansyah, 2013). Salah satu cara penulisan algoritma Dijkstra dapat dilihat pada tabel 2.4 sebagai berikut.

Tabel 2.4 Algoritma Dijkstra dalam bentuk *pseudocode*
(Munir, 2005 hal.414)

<pre>procedure Dijkstra (INPUT m : matriks dan a : simpul awal)</pre>
<p>(Mencari lintasan terpendek dari simpul awal a ke semua simpul lainnya Masukan : matriks ketetanggaan (m) dari graf berbobot G dan simpul awal a Keluaran : lintasan terpendek dari a ke semua simpul lainnya)</p>
<p><u>Deklarasi</u></p> <p>s1, s2, ..., sn :integer (tabel integer) d1, d2, ..., dn :integer (tabel integer) i, j, k: integer</p>
<p><u>Algoritma (langkah 0 / Inisialisasi)</u></p> <p>for i \leftarrow 1 to n do $s_i \leftarrow 0$ $d_i \leftarrow m_{ai}$ Endfor</p>
<p><u>Algoritma (langkah 1)</u></p> <p>Sa \leftarrow 1 (karena simpul a adalah simpul asal lintasan terpendek, jadi simpul a sudah pasti terpilih dalam lintasan terpendek) Dao0 (tidak ada lintasan terpendek dari simpul a ke a)</p>
<p><u>Algoritma (langkah 2, 3, ..., n-1:)</u></p> <p>For k \leftarrow 2 to n-1 do J \leftarrow simpul dengan sj = 0 dan dj minimal $S_j \leftarrow 1$ {simpul j sudah terpilih ke dalam lintasan terpendek} {perbarui tabel d} For semua simpul I dengan si= 0 do If dj+mji < di then Didj+mji Endif Endfor Endfor)</p>

2.4.2 Algoritma Bellman Ford

Algoritma Bellman Ford merupakan algoritma yang dikembangkan oleh Richard Bellman dan Lester Ford. Algoritma ini digunakan untuk mencari lintasan terpendek dengan sumber tunggal (Purwanto, 2008). Perbedaannya dengan algoritma Dijkstra adalah algoritma Bellman mampu untuk memiliki bobot yang negatif yang dapat menjadi kelebihan dari beberapa kasus tertentu. Diberikan sebuah graf yang berbobot dan memiliki arah $G = (V,E)$ dengan simpul awal s . Lalu fungsi bobot $w : E \rightarrow R$. Algoritma Bellman mengembalikan sebuah nilai *boolean* yang akan mengindikasikan apakah terdapat siklus berbobot negatif yang dapat dilalui oleh simpul awal. Jika didapati keadaan demikian, maka algoritma akan mengindikasikan tidak terdapat solusi *shortest path*. Sebaliknya, jika tidak didapati keadaan demikian, maka algoritma akan menghasilkan *shortest path* beserta bobotnya. Algoritma Bellman Ford ini dapat didefinisikan sebagai berikut (Cormen, 1990) , perhatikan tabel 2.5.

Tabel 2.5 Algoritma Bellman Ford

(Cormen, 1990, 532-533)

Algoritma Bellman Ford (G, w, s)

1. Initialize-Single-Source(G,s)
2. **for** $I \leftarrow 1$ to $|V[G]| - 1$
3. **do for** setiap sisi $(u, v) \in E[G]$
4. **do Relax** (u,v,w)
5. **for** setiap sisi $(u, v) \in E[G]$
6. **do if** $d[v] > d[u] + w(u, v)$
7. **then return** FALSE
8. **Return** TRUE

Kelemahan algoritma ini adalah sedikit lambat dan dapat menyebabkan keterlambatan (*delay*) antar *router*, sehingga dapat menyebabkan waktu dan sumberdaya jaringan terbuang.

2.4.3 Algoritma Floyd Warshall

Algoritma Flyod Warshall seperti beberapa algoritma yang telah disebutkan sebelumnya merupakan algoritma yang mencari lintasan terpendek. Algoritma ini dapat mencari semua jarak dari tiap simpul. Hal ini berarti algoritma ini juga dapat digunakan untuk menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah dan seluruh pasangan titik secara langsung membuat tabel lintasan terpendek antar simpul (Purwanato, 2005). Algoritma Floyd merupakan varian pemrograman dinamis, artinya solusi-solusi tersebut dibentuk dari solusi yang berasal dari tahap sebelumnya dan memiliki kemungkinan jumlah solusi lebih dari satu (Fauzi, 2011). Proses kerja dari algoritma Floyd adalah sebagai berikut, perhatikan tabel 2.6.

Tabel 2.6 Algoritma Floyd Warshall

(Sagih, 2014)

Algoritma Floyd : Proses perhitungan <i>shortest path</i>
Floyd-Warshall(d,n)
<pre> 1. for k ←1 to n 2. for j ←1 to n 3. for i ←1 to n 4. $d_{ij}(k) = \min(d_{ij}(k-1), d_{ij}(k-1) + d_{ij}(k-1))$ 5. if choose $d_{ij}(k-1) + d_{ij}(k-1)$ then pred(ij) = k </pre>
Algoritma Floyd : Proses penentuan rute <i>shortest path</i>
Path (I, j) <ol style="list-style-type: none"> If pred(I, j) = nil then Return output(I, j) Else

- | | |
|----|---------------------|
| 4. | Path(i, pred(I, j)) |
| 5. | Path(pred(I, j), j) |

2.4.4 Perbandingan Algoritma Dijkstra, Bellman Ford, dan Floyd Warshall

Perbandingan Algoritma Dijkstra, Bellman Ford, dan Floyd Warshall atas beberapa faktor seperti jenis algoritma, prinsip cara kerja, jenis pemrograman, kompleksitas waktu, dan nilai bobot simpul. Perhatikan tabel 2.7.

**Tabel 2.7 Perbandingan algoritma pencarian jarak terpendek
(shortest path algorithm) (Fauzi, 2012)**

	Dijkstra Algorithm	Bellman Ford Algorithm	Floyd Warshall Algorithm
Jenis Algoritma	<i>Single Source Shortest Path</i>	<i>Pairs Shortest Path</i>	<i>All Pair Shortest Path</i>
Prinsip cara kerja	Priority Queue	Priority Queue	Matrix
Jenis Pemrograman	Greedy	Dinamis	Dinamis
Kompleksitas Waktu (time complexity)	$O(E + V \log V)$	$O(VE)$	$O(n^3)$
Kompleksitas Ruang (space complexity)	$O(v^2)$	$O(v^2)$	$O(v^3)$
Nilai bobot simpul	Positif	Positif, Negatif	Positif, Negatif

Dalam tabel ini, tabel 2.7, terlihat bahwa kompleksitas waktu yang dimiliki algoritma dijkstra memiliki kompleksitas yang lebih kecil. Penelitian ini akan menggunakan algoritma Dijkstra untuk diimplementasikan ke dalam perancangan aplikasi OAM.

Karena algoritma Dijkstra menggunakan prinsip Greedy, hal itu membuat algoritma dijkstra lebih cocok jika digunakan dalam suatu jaringan karena algoritma tersebut dapat mengetahui konfigurasi keseluruhan jaringan dengan kebutuhan waktu (*running time*) yang lebih kecil (Irawan, 2011). Algoritma Dijkstra tidak memiliki bobot negatif, sehingga sesuai dengan kriteria penelitian yang tidak memerlukan pencarian terhadap nilai negatif. Dalam kasus terburuk, algoritma Dijkstra tetap dapat menemukan solusi terbaik dengan kompleksitas yang lebih rendah dibandingkan algoritma Bellman Ford dan algoritma Floyd.

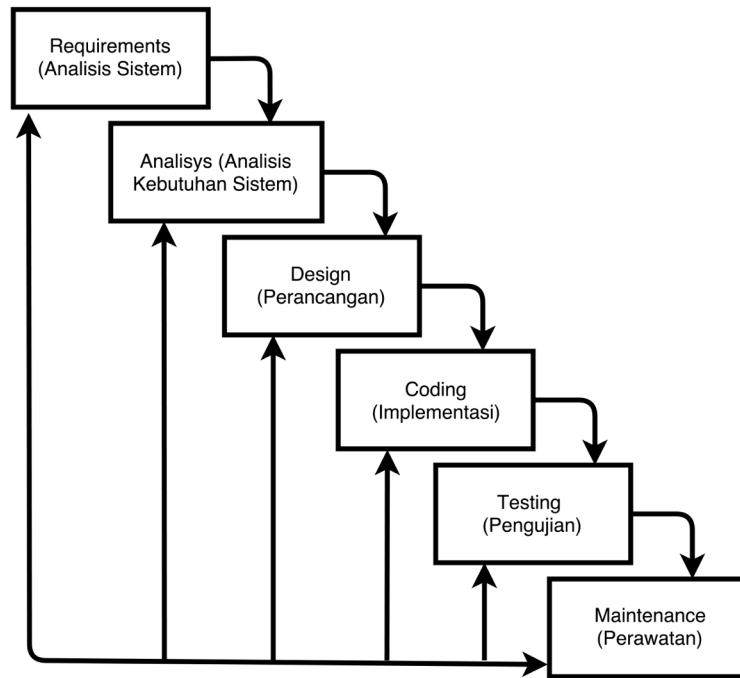
2.5 *System Development Life Cycle (SDLC)*

System Development Life Cycle atau SDLC atau Siklus Hidup Pengembangan Sistem adalah metode pengembangan sistem tradisional yang digunakan sebagian besar organisasi saat ini. SDLC adalah kerangka kerja *framework* yang terstruktur yang berisi proses - proses sekuensial di mana sistem informasi dikembangkan (Turban, 2003). *System Development Life Cycle* (SDLC) merupakan salah satu metode pengembangan sistem informasi yang popular pada saat sistem informasi pertama kali dikembangkan (Susanto, 2004). Model yang ini biasanya disebut juga sebagai model *waterfall* atau *classic life cycle*. Hal ini dikarenakan bentuknya yang bertingkat ke bawah dari satu fase ke fase lainnya, model ini dikenal dengan model waterfall, seperti terlihat pada Gambar 2.6. Tahapan-tahapannya adalah Requirements (Analisis Sistem), Analysis (Analisis Kebutuhan Sistem), Design (Perancangan), Coding (Implementasi), Testing (Pengujian) dan Maintenance (Perawatan) (Pressman, 1997). Penjelasan metode *System Development Life Cycle* (SDLC) adalah sebagai berikut:

1) *Requirements* (Analisis Sistem)

Taham ini mengumpulkan ide-ide aplikasi dari yang dibutuhkan aplikasi , kemudian penulis (*developer*) menganalisa kebutuhan – kebutuhan tersebut untuk membangun sebuah aplikasi. Jika aplikasi yang dijelaskan sebelumnya

sudah ada di lingkungan masyarakat, maka *developer* akan melakukan *brainstorming* untuk meningkatkan performa dari aplikasi tersebut.



Gambar 2.6 Diagram proses pada *System Development Life Cycle (SDLC) model Waterfall* (Pressman, 1977)

Aplikasi yang bersifat baru akan didokumentasi untuk membantu pengembangan selanjutnya sedangkan aplikasi yang bersifat sama dengan yang terdapat di lingkungan masyarakat akan dilakukan studi komparasi kemudian dilakukan pengembangan. Tahap ini bertujuan untuk mengumpulkan kebutuhan-kebutuhan pengguna dan kemudian mentransformasikan ke dalam sebuah deskripsi yang jelas dan lengkap.

2) *Analisis* (Analisis Kebutuhan Sistem)

Di dalam tahap ini penulis (*developer*) perlu mengetahui karakteristik dasar aplikasi yang akan dibangun. Hal ini meliputi fungsi, bentuk, serta tampilannya. Tahap ini menghasilkan arsitektur dan *storyboard* dari tampilan aplikasi. Ketika rancangan fungsi dan tampilan selesai, seluruh rancangan yang telah dikumpulkan akan di dokumentasi untuk dilanjutkan ke tahap selanjutnya yaitu tahap *design*. Tahap analisis sistem ini bertujuan untuk menjabarkan segala sesuatu yang nantinya akan ditangani oleh perangkat lunak. Tahapan ini adalah tahapan pemodelan yang merupakan sebuah representasi obyek di dunia nyata.

3) *Design* (Perancangan)

Setelah melakukan analisis sistem (*requirements*) dan analisis kebutuhan sistem (*analisis*), selanjutnya kita akan memasuki tahapan perancangan (*design*). Untuk membuat suatu aplikasi yang baik diperlukan merancang kebutuhan struktur data, arsitektur perangkat lunak, detil prosedur dan karakteristik tampilan yang akan disajikan secara khusus. Proses ini menterjemahkan kebutuhan aplikasi ke dalam sebuah model aplikasi perangkat lunak yang dapat diperhatikan kualitasnya sebelum melanjutkan ke tahap implementasi.

4) *Coding* (Implementasi)

Rancangan yang telah dibuat dalam tahap perancangan (*design*) akan diterjemahkan ke dalam suatu bentuk atau bahasa yang dapat dibaca dan diterjemahkan oleh komputer untuk diolah. Tahap implementasi (*coding*) merupakan tahap yang mengkonversi apa yang telah dirancang sebelumnya ke dalam sebuah bahasa yang dimengerti oleh logika dan bahasa komputer. Kemudian komputer akan menjalankan fungsi-fungsi yang telah didefinisikan

oleh penulis (*developer*) sehingga mampu menciptakan aplikasi yang berjalan dengan baik. Tahap ini merupakan tahap pemrograman aplikasi yang merealisasikan apa yang telah direncanakan pada tahap perancangan (*design*).

5) *Testing* (Pengujian)

Pengujian secara fungsional pada *coding* aplikasi yang telah dilakukan pada tahap implementasi di lakukan pada fase ini. Pengujian program aplikasi dilakukan untuk mengetahui apabila terjadi kesalahan pada program yang telah dibuat, serta memastikan proses *input* berjalan dengan benar, sehingga dapat menghasilkan *output* yang sesuai ekspektasi. Tahap ini terdapat 2 metode pengujian perangkat yang dapat digunakan, yaitu metode *Black Box* dan metode *White Box*. Pengujian menggunakan metode *black box* menekankan pengujian fungsionalitas dari aplikasi tanpa harus mengetahui bagaimana struktur di dalam perangkat lunak tersebut. Pengujian *black box* dikatakan berhasil jika fungsi-fungsi yang ada telah memenuhi spesifikasi kebutuhan yang telah dibuat sebelumnya. Pengujian dengan menggunakan metode *white box* berfokus menguji struktur internal perangkat lunak dengan melakukan pengujian pada algoritma yang digunakan oleh aplikasi.

6) *Maintenance* (Perawatan)

Fase ini merupakan fase terakhir pada SDLC. Pada tahap ini, pengguna (*user*) aplikasi akan diberikan (ujicoba) menjalankan aplikasi untuk melihat penyesuaian dan perubahan fungsi yang sesuai keadaan yang diinginkan / diperlukan. Perawatan dan perbaikan aplikasi diperlukan sewaktu waktu untuk memperbaiki kekurangan dan menambah fitur – fitur aplikasi baru yang dirasa perlu. *Feedback* dari pengguna aplikasi yang telah mengunduh / menggunakan aplikasi akan terkumpulkan pada fase ini. Tahap ini juga merupakan tahapan untuk memperbaiki *bugs* yang ada dan *updates*.

BAB III

METODE PENELITIAN

3.1 Perancangan Aplikasi

Perancangan aplikasi akan menerapkan metode SDLC yang seperti telah dijelaskan pada BAB II. SDLC adalah kerangka kerja *framework* yang terstruktur yang berisi proses - proses sekuensial di mana sistem informasi dikembangkan (Turban, 2003).

1) *Requirements* (Analisis Sistem)

Tahap ini merupakan tahap pertama untuk menentukan ide penelitian. Tahap disertai dengan observasi pendapat dan penelitian dari para peneliti – peneliti terdahulu, menganalisa dari berbagai sumber yang *legitimate* secara *online* maupun *offline*, seperti membaca *paper*, jurnal dan sumber lainnya untuk mengembangkan aplikasi yang sudah pernah dibuat oleh pembuat sebelumnya. Aplikasi yang akan dibangun, yaitu *Order Ambulance Mobile* (OAM), merupakan aplikasi yang berbeda dibandingkan dengan beberapa peneliti – peneliti sebelumnya karena aplikasi OAM bergerak di bidang kesehatan, menggunakan bahasa pemrograman android, dan menggunakan algoritma Dijkstra. Tahap ini telah dilakukan pada BAB I dan BAB II. Tahap selanjutnya adalah analisis kebutuhan sistem (*analisisys*).

2) *Analisisys* (Analisis Kebutuhan Sistem)

Tahap kedua dari SDLC merupakan analisis kebutuhan sistem (*analisisys*). Tahap ini merupakan tahap analisis kebutuhan sistem untuk membangun aplikasi OAM. Pada tahap analisis ini ada beberapa spesifikasi perangkat yang dibutuhkan dalam pengembangan aplikasi, yaitu :

A. Kebutuhan Perangkat Keras

Kebutuhan perangkat keras yang dibutuhkan dalam pembangunan maupun pengoperasian sistem aplikasi ini yaitu menggunakan Smartphone Android dan Komputer dengan spesifikasi seperti berikut:

Smartphone Android (Client)

1. O.S Smartphone : Kitkat 4.4.2
2. Nomer Model : G900I
3. Nama Smartphone : Samsung Galaxy S5
4. Processor : Quad core Krait 400 2.5 Ghz
5. Camera : 16 Megapixel

Komputer (Server)

1. Nama Komputer : Asus N46VN
2. Processor : Pentium Core i5 2.5GHz
3. RAM : 4 GByte
4. VGA : nVidia Geforce GT 630M 2GByte
5. Hard Disk : 750 Gb
6. Monitor : 14" screen

B. Kebutuhan Perangkat Lunak

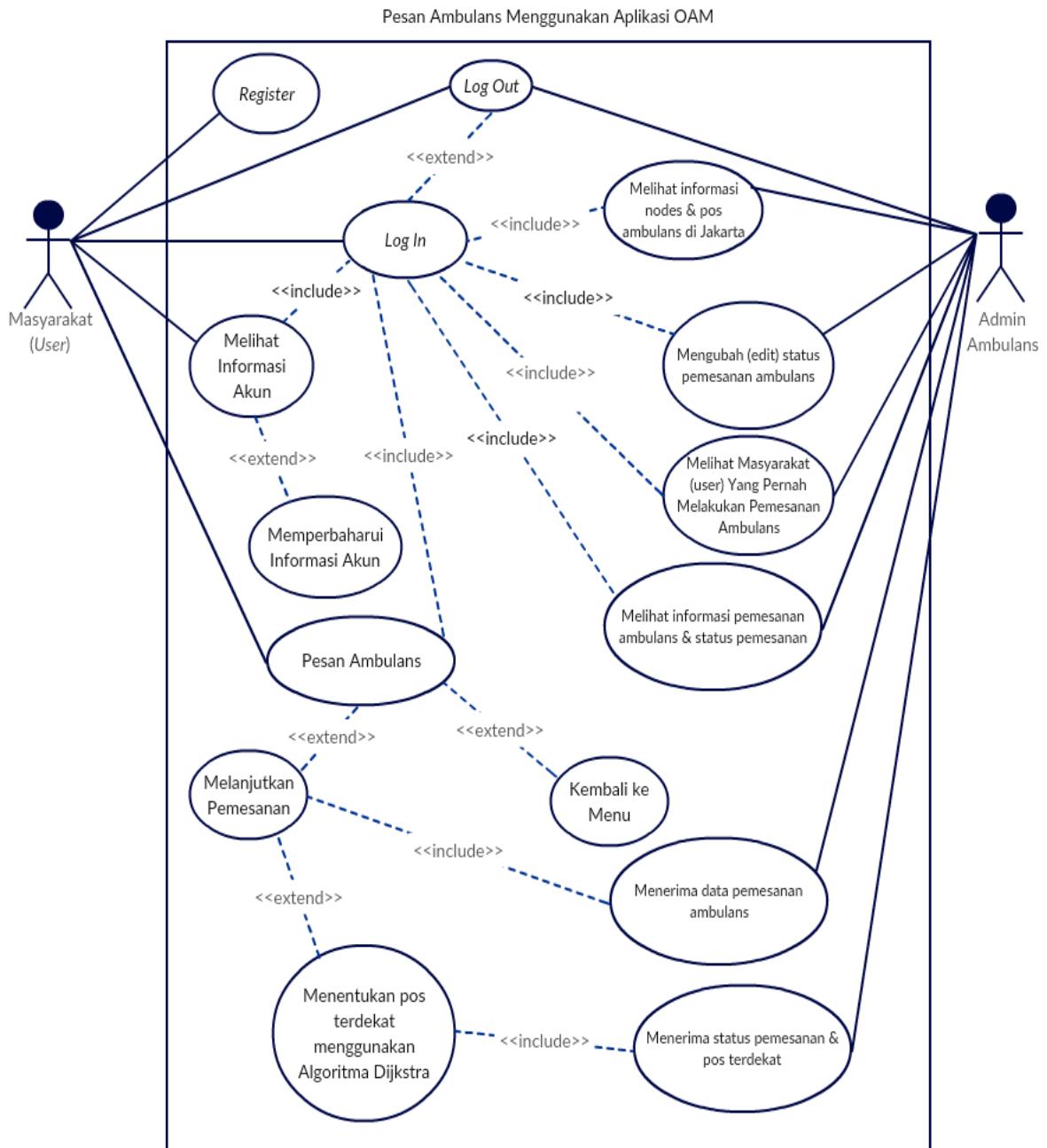
Kebutuhan perangkat lunak adalah perangkat lunak yang dibutuhkan dalam pembangunan sistem aplikasi ini, diantaranya sebagai berikut

1. Eclipse Indigo
2. Android SDK
3. Adobe Photoshop
4. Adobe Illustration
5. Adobe Experience
6. PHP PDT
7. MySQL
8. WebHosting
9. Gmap3
10. Template parser omap-ci
11. Web Browser
12. Twitter Bootstrap 2.2.1
13. CodeIgniter 2.1.3
14. Web Browser

3) *Design* (Perancangan)

Tahap ini merupakan hasil observasi dari beberapa sumber untuk pengumpulan ide dan wawasan pada tahap *requirements* (analisis sistem) dan *analisis* (analisa kebutuhan sistem). Pada tahap ini akan dibuat desain dari hasil identifikasi pada tahap sebelumnya. Desain alur dan cara kerja aplikasi akan direpresentasikan dengan UML diagram yaitu *use case diagram*, *sequence diagram*, *activity diagram* dan *class diagram*. Untuk desain tampilan aplikasi akan diletakkan pada bagian *user interface*.

a. Use case diagram



Gambar 3.1 Use Case Diagram Aplikasi OAM

Use case diagram yang digunakan pada aplikasi OAM dapat dilihat pada gambar 3.1. Berikut merupakan deskripsi dari *use case* diagram aplikasi OAM (gambar 3.1) dari tabel 3.1 hingga tabel 3.17:

Tabel 3.1 Deskripsi Use Case Register

Use Case Name	Register	
Use Case ID	1	
Actor	Masyarakat (<i>user</i>)	
Description	<i>Use Case</i> ini merupakan pendaftaran akun yang dilakukan oleh masyarakat (<i>user</i>) untuk mendapat akses pemesanan ambulans melalui aplikasi.	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM.	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) melakukan pendaftaran terlebih dahulu pada saat aplikasi OAM dibuka sebelum melakukan pemesanan ambulans.	
Typical Of Events	Actor Action	System Response
	Membuka Aplikasi OAM.	Menampilkan halaman pendaftaran (<i>register</i>) dan halaman <i>log in</i> .
Alternate Course	-	
Post - Condition	Yang ditampilkan pada aplikasi OAM di <i>smartphone</i> masyarakat (<i>user</i>) adalah <i>text field</i> untuk alamat <i>email</i> , <i>password</i> , nama lengkap, dan nomor hp, serta tombol (<i>button</i>) <i>register</i> .	

Tabel 3.2 Deskripsi Use Case Log In

Use Case Name	Log In	
Use Case ID	2	
Actor	Masyarakat (<i>user</i>)	
Description	Masyarakat (<i>user</i>) dapat masuk akun aplikasi <i>Order Ambulance Mobile</i> (OAM) dari <i>use case</i> ini	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM.	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) memasukkan alamat <i>email</i> dan <i>password</i> terlebih dahulu pada saat aplikasi OAM dibuka sebelum melakukan pemesanan ambulans.	

<i>Typical Of Events</i>	<i>Actor Action</i>	<i>System Response</i>
	Membuka Aplikasi OAM.	Menampilkan halaman <i>log in</i> dan halaman pendaftaran (<i>register</i>). Yang ditampilkan pada aplikasi OAM di <i>smartphone</i> masyarakat (<i>user</i>) adalah <i>text field</i> untuk alamat <i>email</i> dan <i>password</i> , serta tombol (<i>button</i>) <i>login</i> .
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	Aplikasi menampilkan halaman menu utama.	

Tabel 3.3 Deskripsi *Use Case Log In*

<i>Use Case Name</i>	<i>Log In</i>	
<i>Use Case ID</i>	3	
<i>Actor</i>	Admin Ambulans	
<i>Description</i>	Admin Ambulans dapat masuk <i>Order Ambulance Mobile</i> (OAM) <i>website</i> dari <i>use case</i> ini	
<i>Pre-Condition</i>	Admin Ambulans membuka <i>website</i> OAM.	
<i>Trigger</i>	<i>Use case</i> ini dilakukan agar admin ambulans memasukkan alamat <i>email</i> dan <i>password</i> terlebih dahulu sebelum melakukan manajemen pemesanan ambulans.	
<i>Typical Of Events</i>	<i>Actor Action</i>	<i>System Response</i>
	Membuka <i>website</i> OAM.	Menampilkan halaman <i>log in</i> dan halaman pendaftaran (<i>register</i>). Yang ditampilkan pada aplikasi OAM <i>website</i> adalah <i>text field</i> untuk alamat <i>email</i> dan <i>password</i> , serta tombol (<i>button</i>) <i>login</i> .
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	Aplikasi menampilkan halaman menu utama.	

Tabel 3.4 Deskripsi Use Case Log Out

Use Case Name	<i>Log Out</i>	
Use Case ID	4	
Actor	Masyarakat (<i>user</i>)	
Description	Masyarakat (<i>user</i>) dapat keluar dari akun aplikasi <i>smartphone Order Ambulance Mobile</i> (OAM) dari <i>use case</i> ini.	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat keluar dari akun aplikasi OAM.	
Typical Of Events	Actor Action	System Response
	Klik “ <i>Log Out</i> ”	Menampilkan halaman <i>log in</i> dan halaman pendaftaran (<i>register</i>) karena masyarakat (<i>user</i>) telah keluar dari akunnya.
Alternate Course	-	
Post - Condition	Aplikasi menampilkan halaman <i>log in</i> dan <i>register</i>	

Tabel 3.5 Deskripsi Use Case Log Out

Use Case Name	<i>Log Out</i>	
Use Case ID	5	
Actor	Admin Ambulans	
Description	Admin Ambulans dapat masuk <i>Order Ambulance Mobile</i> (OAM) website dari <i>use case</i> ini	
Pre-Condition	Admin Ambulans membuka website OAM.	
Trigger	<i>Use case</i> ini dilakukan agar admin ambulans dapat keluar dari akun website OAM.	
Typical Of Events	Actor Action	System Response
	Klik “ <i>Log Out</i> ”	Menampilkan halaman <i>log in</i> karena admin ambulans sudah keluar dari akunnya.
Alternate Course	-	

Post - Condition	Aplikasi menampilkan halaman <i>log in</i> .
-------------------------	--

Tabel 3.6 Deskripsi Use Case Melihat Informasi Akun

Use Case Name	Melihat Informasi Akun	
Use Case ID	6	
Actor	Masyarakat (<i>user</i>)	
Description	Pada aplikasi <i>smartphone Order Ambulance Mobile</i> (OAM), masyarakat (<i>user</i>) dapat melihat informasi akun berupa nama, <i>email</i> dan nomor hp	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat mengetahui informasi akun aplikasi OAM.	
Typical Of Events	Actor Action	System Response
	Klik “Lihat Info Akun”	Menampilkan informasi akun berupa nama, <i>email</i> dan nomor hp.
Alternate Course	-	
Post - Condition	Aplikasi menampilkan halaman info akun	

Tabel 3.7 Deskripsi Use Case Memperbarui Informasi Akun

Use Case Name	Memperbarui Informasi Akun	
Use Case ID	7	
Actor	Masyarakat (<i>user</i>)	
Description	Pada aplikasi <i>smartphone Order Ambulance Mobile</i> (OAM), masyarakat (<i>user</i>) dapat memperbarui informasi akun berupa nama, nomor hp, <i>email</i> dan <i>password</i> .	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat memperbarui / <i>update</i> informasi akun aplikasi OAM.	
Typical Of Events	Actor Action	System Response
	Klik “Lihat Info Akun”	Menampilkan informasi akun berupa nama, nomor hp, <i>email</i> dan <i>password</i> .
Alternate Course	-	

Post - Condition	Aplikasi menampilkan halaman info akun
-------------------------	--

Tabel 3.8 Deskripsi Use Case Pesan Ambulans

Use Case Name	Pesan Ambulans	
Use Case ID	8	
Actor	Masyarakat (<i>user</i>)	
Description	Pada aplikasi <i>smartphone Order Ambulance Mobile</i> (OAM), masyarakat (<i>user</i>) dapat melakukan pemesanan ambulans setelah melakukan <i>log in</i> terlebih dahulu.	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat melakukan pemesanan ambulans	
Typical Of Events	Actor Action	System Response
	Klik “Pesan Ambulans”	Menampilkan <i>textfield</i> untuk memasukkan beberapa informasi diantaranya nama jalan, patokan alamat, dan deskripsi penyakit pasien.
Alternate Course	-	
Post - Condition	Aplikasi menampilkan halaman pemesanan ambulans	

Tabel 3.9 Deskripsi Use Case Melanjutkan Pemesanan

Use Case Name	Melanjutkan Pemesanan	
Use Case ID	9	
Actor	Masyarakat (<i>user</i>)	
Description	<i>Use case</i> ini ditemukan setelah masyarakat (<i>user</i>) masuk ke <i>use case</i> pesan ambulans. <i>Use case</i> ini melanjutkan proses pemesanan ambulans dengan cara mengirimkan data pemesanan ambulans berupa nama jalan, patokan jalan, dan deskripsi penyakit ke <i>web service</i> OAM.	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat melakukan pemesanan ambulans	

<i>Typical Of Events</i>	<i>Actor Action</i>	<i>System Response</i>
	Klik “Lanjutkan Pemesanan”	Menampilkan beberapa titik nodes rute, titik lokasi pos dan titik lokasi pemesan. System mengirimkan informasi nama jalan, patokan alamat, dan deskripsi penyakit pasien ke <i>web service</i> OAM.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	Aplikasi menampilkan halaman pemesanan ambulans	

Tabel 3.10 Deskripsi Use Case Menentukan Pos Terdekat**Menggunakan Algoritma Dijkstra**

<i>Use Case Name</i>	Menentukan Pos Terdekat Menggunakan Algoritma Dijkstra	
<i>Use Case ID</i>	10	
<i>Actor</i>	Masyarakat (<i>user</i>)	
<i>Description</i>	<i>Use case</i> ini ditemukan setelah masyarakat (<i>user</i>) masuk ke <i>use case</i> melanjutkan pemesanan ambulans. Logika Algoritma Dijkstra digunakan pada <i>use case</i> ini untuk menentukan pos ambulans terdekat dari titik pemesan dan mengirimkannya ke <i>web service</i> OAM.	
<i>Pre-Condition</i>	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
<i>Trigger</i>	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat melakukan pemesanan ambulans	
<i>Typical Of Events</i>	<i>Actor Action</i>	<i>System Response</i>
	Klik “Find Path”	Menampilkan beberapa titik nodes rute, titik lokasi pos dan titik lokasi pemesan. Aplikasi akan mendapatkan hasil akhir dari pencarian pos terdekat, lalu dan akan mengirimkan informasi pencarian ke <i>web service</i> OAM.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	Aplikasi menampilkan halaman pemesanan ambulans	

Tabel 3.11 Deskripsi Use Case Kembali ke Menu Utama

Use Case Name	Kembali ke Menu Utama	
Use Case ID	11	
Actor	Masyarakat (<i>user</i>)	
Description	Masyarakat (<i>user</i>) yang telah melalui <i>use case</i> pesan ambulans dapat membatalkan proses pemesanan dengan cara kembali ke menu awal aplikasi.	
Pre-Condition	Masyarakat (<i>user</i>) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar masyarakat (<i>user</i>) dapat membatalkan pemesanan ambulans	
Typical Of Events	Actor Action	System Response
	Klik “Kembali”	Menampilkan halaman menu utama aplikasi OAM.
Alternate Course	-	
Post - Condition	Aplikasi menampilkan halaman menu utama aplikasi OAM.	

Tabel 3.12 Deskripsi Use Case Menerima Data Pemesanan Ambulans

Use Case Name	Menerima Data Pemesanan Ambulans	
Use Case ID	12	
Actor	Admin Ambulans	
Description	Admin ambulans menerima data pemesanan ambulans yang dikirimkan oleh masyarakat (<i>user</i>) melalui <i>smartphone</i> mereka, informasi pemesanan ambulans meliputi nama jalan, patokan jalan, dan deskripsi penyakit melalui <i>use case</i> ini.	
Pre-Condition	Admin ambulans membuka <i>website</i> OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar admin ambulans mendapatkan informasi pemesanan ambulans	
Typical Of Events	Actor Action	System Response
	-	Web service OAM akan secara otomatis mendapatkan informasi

		pemesanan ambulans berupa nama jalan, patokan jalan, dan deskripsi ketika masyarakat (<i>user</i>) klik lanjutkan pemesanan pada aplikasi OAM.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>		<i>Web service OAM mendapatkan informasi pemesanan ambulans</i>

Tabel 3.13 Deskripsi Use Case Menerima Pemesanan Pos Terdekat

Use Case Name	Menerima Status Pemesanan Pos Terdekat	
Use Case ID	13	
Actor	Admin Ambulans	
Description	Admin ambulans menerima data pos pemesanan ambulans terdekat, dalam radius 3 km, dari titik pesanan yang dikirimkan oleh masyarakat (<i>user</i>) melalui <i>smartphone</i> mereka pada <i>use case</i> ini.	
Pre-Condition	Admin ambulans membuka <i>website OAM</i> dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar admin ambulans mendapatkan informasi pemesanan ambulans	
Typical Of Events	Actor Action	System Response
	-	<i>Web service OAM</i> akan secara otomatis mendapatkan informasi pemesanan ambulans berupa status <i>handled</i> atau <i>out of coverage</i> dan pos ambulans terdekat.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	<i>Web service OAM</i> mendapatkan informasi pemesanan ambulans	

**Tabel 3.14 Deskripsi Use Case Melihat Informasi Titik Rute / Nodes
dan Pos Ambulans di Jakarta**

Use Case Name	Melihat Informasi Titik Rute / Nodes dan Pos Ambulans di Jakarta
Use Case ID	14

Actor	Admin Ambulans	
Description	Pada aplikasi website <i>Order Ambulance Mobile</i> (OAM), admin ambulans dapat melihat informasi titik – titik pos ambulans di Jakarta dan titik – titik rute (<i>nodes</i>) yang tersedia.	
Pre-Condition	Admin ambulans membuka website OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar admin ambulans mendapatkan informasi titik-titik rute (<i>nodes</i>) dan pos ambulans di Jakarta	
Typical Of Events	Actor Action	System Response
	-	Informasi titik – titik rute / <i>nodes</i> dan pos – pos ambulans dapat dilihat secara langsung pada halaman menu utama website OAM.
Alternate Course	-	
Post - Condition	Web service OAM mendapatkan informasi titik rute nodes dan pos ambulans	

Tabel 3.15 Deskripsi *Use Case* Mengubah (*edit*) Status Pemesanan Ambulans

Use Case Name	Mengubah (<i>edit</i>) Status Pemesanan Ambulans	
Use Case ID	15	
Actor	Admin Ambulans	
Description	Pada aplikasi website <i>Order Ambulance Mobile</i> (OAM), admin ambulans dapat mengubah (<i>edit</i>) status pemesanan ambulans menjadi <i>declined</i> setelah melakukan konfirmasi via telepon terkait kebenaran pemesanan ambulans.	
Pre-Condition	Admin ambulans membuka website OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar admin ambulans dapat mengubah status pemesanan ambulans yang tidak dapat dikonfirmasi kebenarannya menjadi <i>declined</i> .	
Typical Of Events	Actor Action	System Response
	-	Status pemesanan ambulans yang tidak dapat dikonfirmasi

		kebenarannya via telefon dapat diubah (<i>edit</i>) melalui <i>use case</i> ini.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	<i>Website OAM</i> memperbarui status pemesanan ambulans	

Tabel 3.16 Deskripsi *Use Case* Melihat Masyarakat (*User*) Yang Pernah Melakukan Pemesanan Ambulans

<i>Use Case Name</i>	Melihat masyarakat (<i>user</i>) Yang Pernah Melakukan Pemesanan Ambulans	
<i>Use Case ID</i>	16	
<i>Actor</i>	Admin Ambulans	
<i>Description</i>	Pada aplikasi <i>website Order Ambulance Mobile</i> (OAM), admin ambulans dapat melihat informasi masyarakat (<i>user</i>) yang pernah melakukan pemesanan ambulans. Informasinya berupa nama, <i>email</i> , dan nomor hp.	
<i>Pre-Condition</i>	Admin ambulans membuka <i>website OAM</i> dan telah melakukan <i>log in</i> .	
<i>Trigger</i>	<i>Use case</i> ini dilakukan agar admin ambulans dapat mengetahui informasi masyarakat (<i>user</i>) yang pernah melakukan pemesanan ambulans.	
<i>Typical Of Events</i>	<i>Actor Action</i>	<i>System Response</i>
	Klik “User”	<i>Website OAM</i> akan menampilkan informasi masyarakat (<i>user</i>) yang pernah melakukan pemesanan ambulans berupa nama, <i>email</i> , dan nomor hp.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	<i>Website OAM</i> mengetahui informasi masyarakat (<i>user</i>) yang pernah melakukan pemesanan ambulans	

Tabel 3.17 Deskripsi *Use Case* Melihat Informasi Pemesanan Ambulans dan Status Pemesanan

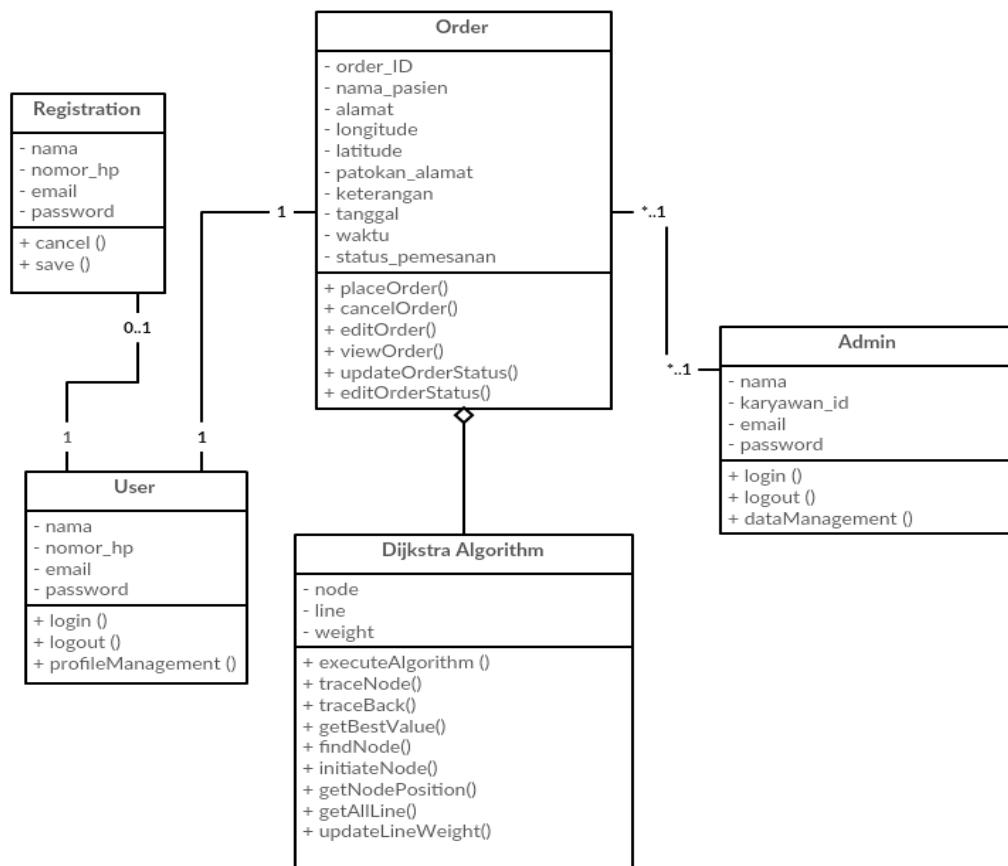
<i>Use Case Name</i>	Melihat Informasi Pemesanan Ambulans dan Status Pemesanan
<i>Use Case ID</i>	17
<i>Actor</i>	Admin Ambulans

Description	Pada aplikasi website <i>Order Ambulance Mobile</i> (OAM), admin ambulans dapat melihat informasi pemesanan ambulans dan status pemesanannya.	
Pre-Condition	Admin ambulans membuka website OAM dan telah melakukan <i>log in</i> .	
Trigger	<i>Use case</i> ini dilakukan agar admin ambulans dapat mengetahui informasi masyarakat (<i>user</i>) yang pernah melakukan pemesanan ambulans.	
Typical Of Events	Actor Action	System Response
	Klik “Order” lalu klik “All”	Website OAM akan menampilkan informasi pemesanan ambulans dan status pemesanan. Informasi berupa tanggal, waktu, alamat, latitude, longitude, status pemesanan, dan pos terdekat.
Alternate Course	-	
Post - Condition	Website OAM mengetahui informasi masyarakat (<i>user</i>) yang pernah melakukan pemesanan ambulans	

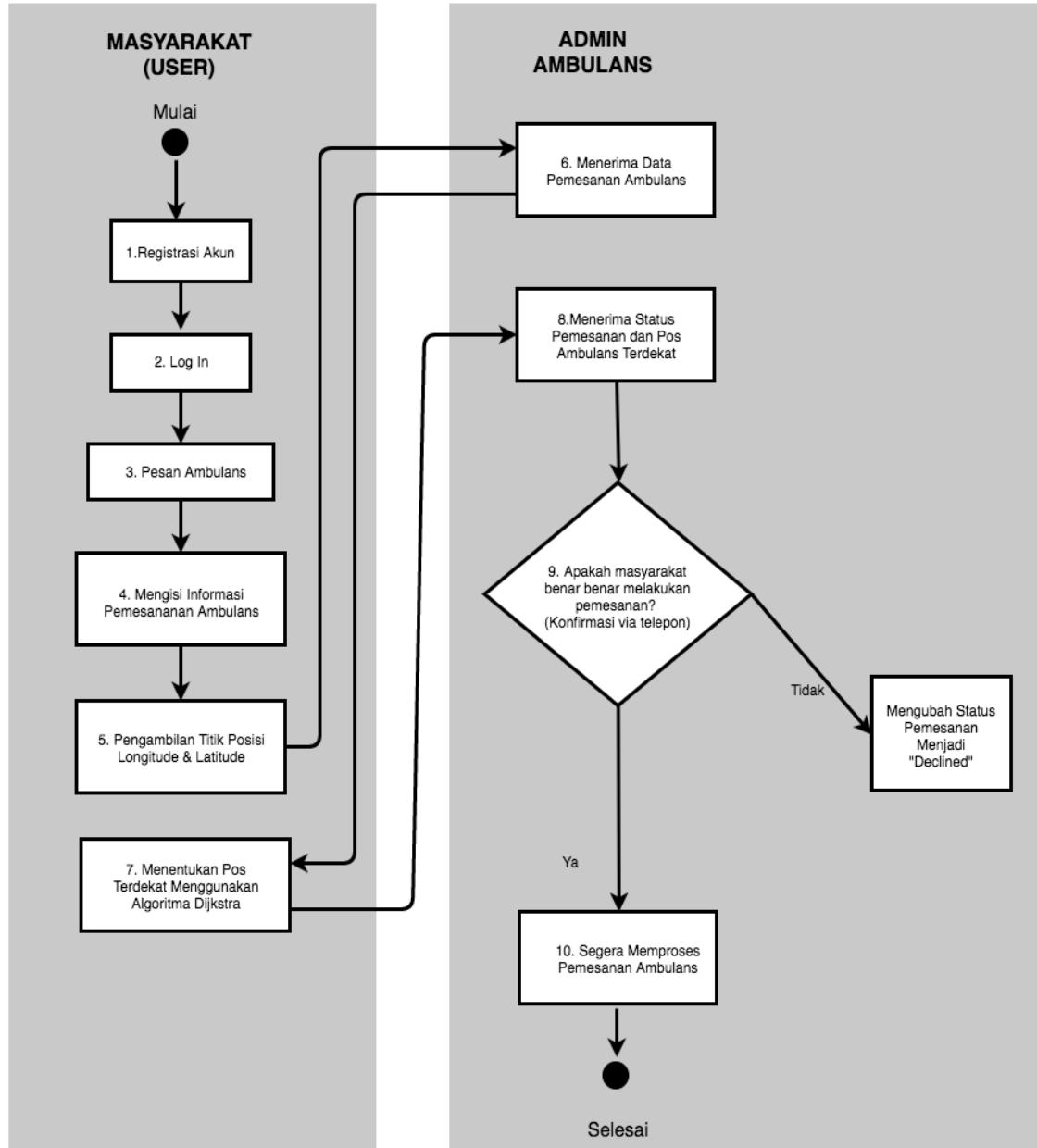
b. *Class Diagram*

Class Diagram merupakan diagram yang menggambarkan sudut sistem dari struktur dan definisi *class* serta hubungannya (*relationship*). *Class* memiliki 3 area pokok, yaitu nama, atribut, dan operasi. Nama berfungsi untuk memberikan identitas pada sebuah *class*. Atribut berfungsi untuk memberikan karakteristik pada data yang dimiliki suatu obyek di dalam *class*. Operasi berfungsi untuk memberikan sebuah fungsi ke sebuah obyek. *Class Diagram* memiliki beberapa simbol yang menjelaskan hubungan (*relationship*) antar *class* yang digunakan pada *class diagram*. *Class Diagram* pada aplikasi OAM memiliki beberapa *class* yaitu *class* masyarakat, *class* pesan ambulans, *class* log in admin, *class* pos ambulans, dan *class* status pemesanan. *Class* masyarakat memiliki beberapa atribut *private* untuk data nama dan nomor hp masyarakat. *Class* masyarakat memiliki hubungan asosiasi berarah (*directed association*) dengan *class* pesan ambulans. *Class* pesan ambulans memiliki beberapa atribut

private untuk data id pemesanan, latitude, longitude, keterangan keadaan, kamera, alamat masyarakat serta deskripsinya. Class pesan ambulans memiliki hubungan dengan class log in admin. Class log in membantu merekap data pemesanan (*issued*) yang telah di *request* oleh masyarakat. Class log in admin memiliki beberapa atribut *private* berupa *username* dan *password*. Class log in admin juga memiliki hubungan dengan class pos ambulans. Class log in admin mencari pos terdekat menggunakan algoritma Dijkstra dari class pos ambulans yang memiliki beberapa atribut *private* berupa data latitude, longitude, alamat pos dan keterangan alamatnya. Class Diagram yang digunakan pada dapat aplikasi *order ambulance mobile* (OAM), terlihat pada gambar 3.2.

Gambar 3.2 *Class Diagram* Aplikasi OAM

c. *Activity Diagram*

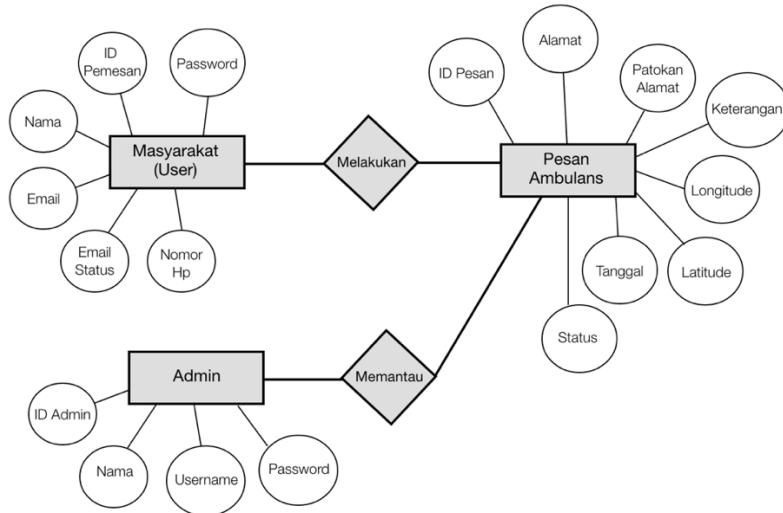


Gambar 3.3 Activity Diagram Aplikasi OAM

Activity diagram yang digunakan pada dapat aplikasi OAM terlihat pada gambar 3.3.

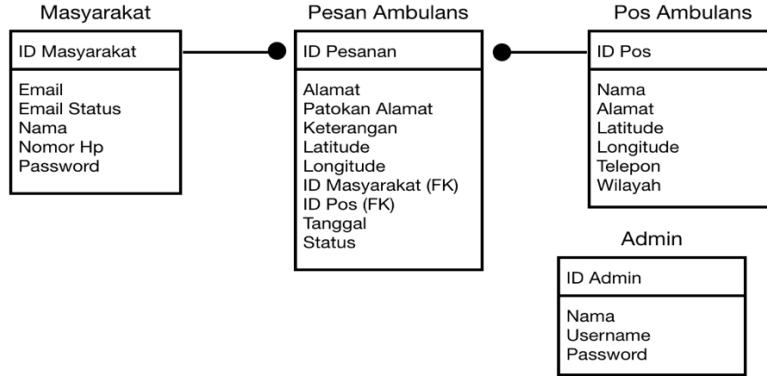
d. Database Design

Sistem ini memiliki dua sisi yaitu di sisi *Server* dan sisi *Client*. Di sisi *Server* ini, yang merupakan *Web Service* aplikasi, digunakan oleh pihak admin ambulans untuk mengolah informasi oleh masyarakat yang telah melakukan pemesanan ambulans melalui aplikasi OAM. Sementara sisi *Client* merupakan aplikasi berbasis android yang digunakan oleh masyarakat sebagai pengguna (*user*) untuk melakukan pemensanan ambulans. Pada gambar 3.4 dapat dilihat bahwa sistem ini memiliki 3 tabel yang diantaranya adalah Tabel Masyarakat, Tabel Pemesanan dan Tabel Admin. Ketiga tabel ini masing masing memiliki attribut dan memiliki hubungan antara satu tabel dengan tabel lainnya (*relationship*).



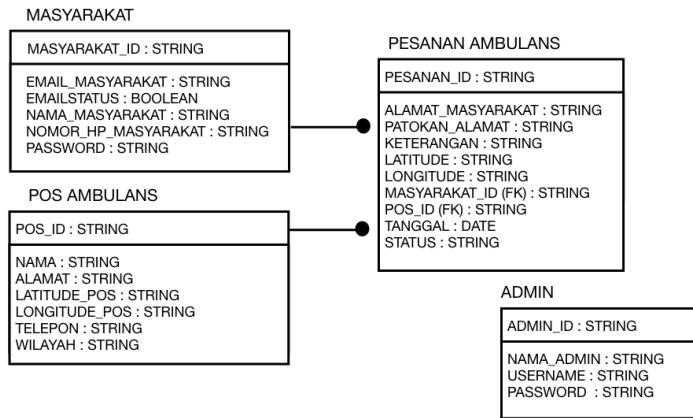
Gambar 3.4 Entity Relationship Diagram (ERD) pada OAM

Logical database model adalah suatu model informasi yang digunakan pada organisasi, instansi, maupun perusahaan berdasarkan pada model data yang spesifik namun tidak tergantung pada *Database Management System* (DBMS) yang khusus dan memiliki pertimbangan fisik lainnya (Connolly, 2002).



Gambar 3.5 Logical Database Model pada OAM

Desain *logical database model* yang akan diimplementasikan pada aplikasi OAM ditampilkan pada Gambar 3.5. *Physical Database Model* adalah konsep bagaimana data akan disimpan pada *storage* dalam bentuk yang menggunakan sejumlah tabel untuk menggambarkan data serta hubungan (*relationship*) antara data-data tersebut. Gambar 3.6 menunjukkan *physical database model* yang akan diimplementasikan pada aplikasi OAM.



Gambar 3.6 Physical Database Model pada OAM

e. User Interface aplikasi OAM

Beberapa contoh tampilan *interface* aplikasi OAM adalah sebagai berikut.

Desain perancangan antarmuka setelah pengguna (*user*) menginstall aplikasi yaitu munculnya halaman untuk melakukan pendaftaran (*register*) pada gambar 3.7, serta halaman *log in* untuk masuk ke aplikasi OAM seperti yang terlihat pada gambar 3.8

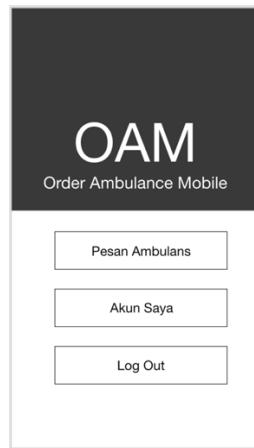
The wireframe shows a registration form. At the top, there are two buttons: 'Log In' on the left and 'Register' on the right, with a horizontal line underneath. Below these buttons is a vertical stack of four input fields, each with a label above it: 'Alamat Email', 'Password', 'Nama Lengkap', and 'Nomor Handphone'. At the bottom of the form is a large rectangular button labeled 'Log In'.

Gambar 3.7 Perancangan Antarmuka pada Aplikasi OAM Bagian Satu

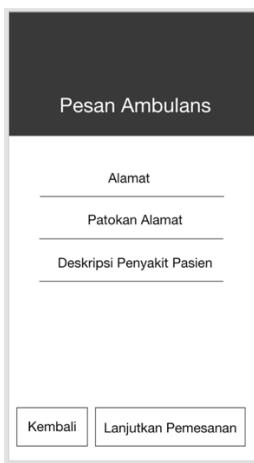
The wireframe shows a login form. At the top, there are two buttons: 'Log In' on the left and 'Register' on the right, with a horizontal line underneath. Below these buttons is a vertical stack of two input fields, each with a label above it: 'Alamat Email' and 'Password'. At the bottom of the form is a large rectangular button labeled 'Log In'.

Gambar 3.8 Perancangan Antarmuka pada Aplikasi OAM Bagian Dua

Desain perancangan antarmuka setelah pengguna (*user*) melakukan *log in* dapat terlihat pada gambar 3.9. Apa bila pengguna (*user*) ingin melakukan pemesanan ambulans menggunakan aplikasi OAM, maka pengguna (*user*) dapat klik “pesan ambulans” lalu halaman untuk pengisian informasi dasar dapat dipenuhi melalui halaman pemesanan ambulans seperti yang terlihat pada gambar 3.10.



Gambar 3.9 Perancangan Antarmuka pada Aplikasi OAM Bagian Tiga



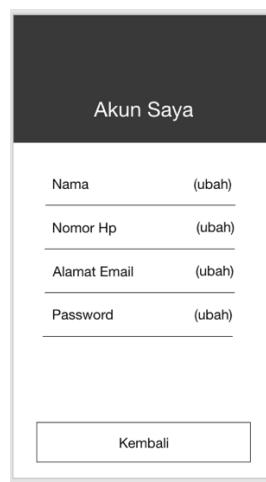
Gambar 3.10 Perancangan Antarmuka pada Aplikasi OAM Bagian Empat

Desain perancangan antarmuka setelah pengguna (*user*) setelah mengisi informasi dasar seperti alamat, patokan alamat, dan deskripsi penyakit pasien

pada gambar 3.10, maka dapat dilanjutkan dengan klik “lanjutkan pemesanan” hingga muncul halaman seperti halaman 3.11. Pada halaman ini user dapat mengetahui titik pos ambulans terdekat dari lokasi pemesan sekaligus melakukan pengiriman data tersebut secara otomatis ke *server* admin. Halaman untuk mengganti informasi akun dapat terlihat pada gambar 3.12 yang diakses pada menu utama sebelumnya seperti yang terlihat pada gambar 3.9



Gambar 3.11 Perancangan Antarmuka pada Aplikasi OAM Bagian Lima



Gambar 3.12 Perancangan Antarmuka pada Aplikasi OAM Bagian Enam

4) *Coding* (Implementasi)

Pada tahap ini rancangan aplikasi yang telah dibuat dalam tahap perancangan (*design*) akan diterjemahkan ke dalam suatu bentuk atau bahasa yang dapat dibaca dan diterjemahkan oleh komputer untuk diolah. Tahap ini merupakan tahap pemrograman aplikasi yang merealisasikan apa yang telah direncanakan pada tahap perancangan (*design*). Hasil implementasi yang dilakukan dalam penelitian ini dapat dilihat pada BAB IV.

5) *Testing* (Pengujian)

Pengujian secara fungsional pada *coding* aplikasi yang telah dilakukan pada tahap implementasi dilakukan pada fase ini. Pengujian program aplikasi dilakukan untuk mengetahui apabila terjadi kesalahan pada program yang telah dibuat, serta memastikan proses *input* berjalan dengan benar, sehingga dapat menghasilkan *output* yang sesuai ekspektasi. Tahap ini terdapat 2 metode pengujian perangkat yang dapat digunakan, yaitu metode *Black Box* dan metode *White Box*. Hasil pengujian aplikasi OAM dalam penelitian ini dapat diliat pada BAB IV.

6) *Maintenance* (Perawatan)

Perawatan serta perbaikan aplikasi diperlukan sewaktu-waktu untuk memperbaiki kekurangan dan menambah fitur – fitur aplikasi baru yang dirasa perlu sesuai keadaan. Pada tahap ini, pengguna (*user*) aplikasi akan diberikan (*ujicoba*) menjalankan aplikasi untuk melihat penyesuaian dan perubahan fungsi yang sesuai keadaan yang diinginkan / diperlukan. *Feedback* dari pengguna aplikasi yang telah mengunduh / menggunakan aplikasi akan terkumpulkan pada fase ini. Tahap ini juga merupakan tahapan untuk memperbaiki *bugs* yang ada dan *updates*.

3.2 Jenis Penelitian

Jenis penelitian pada penelitian ini merupakan *Applied Research* (Penelitian Terapan). *Applied Research* (Penelitian Terapan) merupakan penelitian yang hasilnya akan digunakan untuk membuat suatu keputusan dalam rangka memecahkan persoalan atau menguji hipotesis. Penelitian ini menggunakan algoritma Dijkstra.

3.3 Objek Penelitian

Objek penelitian pada tugas akhir ini adalah aplikasi *Order Ambulance Online* (OAM). Aplikasi OAM merupakan aplikasi di bidang kesehatan yang dapat membantu masyarakat untuk melakukakn pemesanan ambulans di wilayah Jakarta.

BAB IV

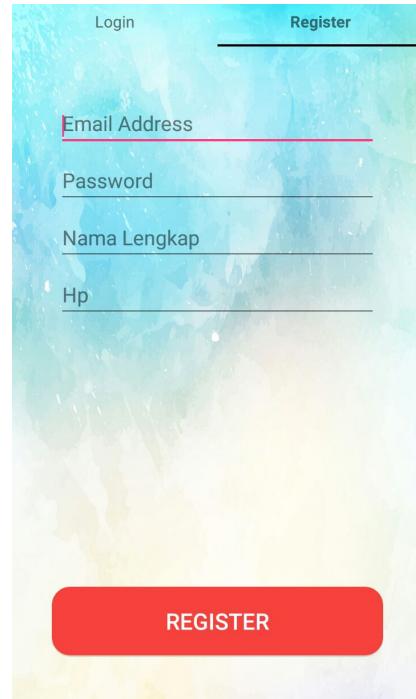
IMPLEMENTASI DAN PENGUJIAN APLIKASI

4.1 Implementasi

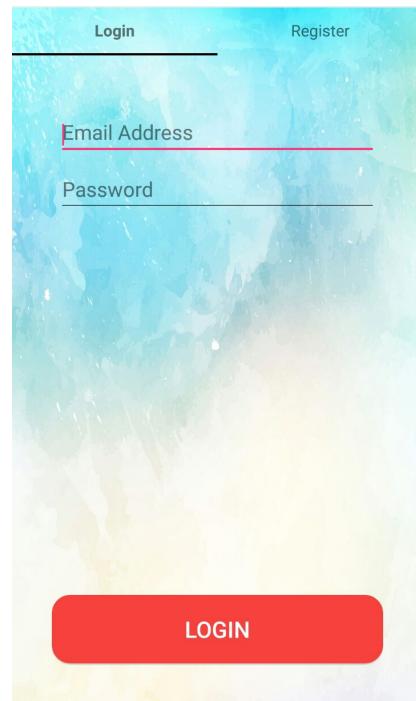
Tahap ini merupakan tahap implementasi (*coding*). Perancangan sistem aplikasi OAM yang telah dibuat pada tahap perancangan (*design*) pada BAB III akan diwujudkan ke dalam bentuk sistem yang sesungguhnya pada bagian ini. Dengan berpatokan pada hasil perancangan, dilakukan pengembangan sistem untuk membangun fitur-fitur sistem agar dapat digunakan oleh masyarakat sebagai pengguna (*user*) dan admin ambulans. Bagian ini akan memaparkan kegiatan implementasi aplikasi OAM pada Android, implementasi aplikasi OAM pada *website Admin*.

4.1.1 Implementasi Aplikasi OAM pada Android

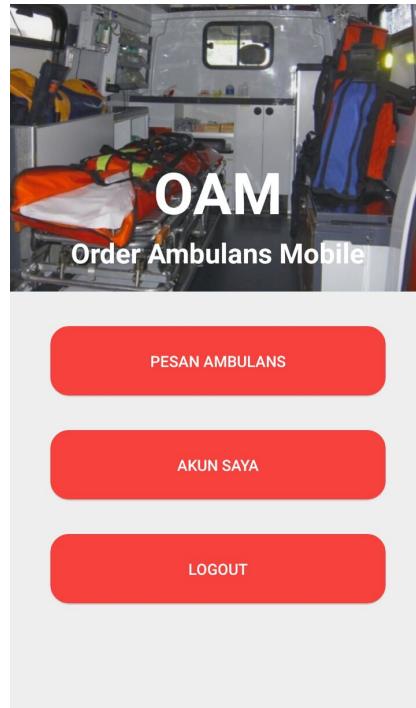
Implementasi pada aplikasi OAM di Android dilakukan berdasarkan perencanaan awal yang telah dilakukan. Perancangan aplikasi OAM sudah dibahas secara lengkap di BAB III pada fase desain aplikasi dari hasil identifikasi sebelumnya. Desain alur dan cara kerja aplikasi akan direpresentasikan dengan UML diagram yaitu *use case diagram*, *sequence diagram*, *activity diagram* dan *class diagram*. Berikut halaman yang muncul setelah menginstall aplikasi OAM. Pada gambar 4.1 terlihat halaman pendaftaran (*register*) untuk pengguna yang baru saja pertama kali mendaftar di aplikasi OAM. Pengguna aplikasi yang baru saja menginstall aplikasi akan diminta mengisi informasi dasar seperti alamat email, password, nama lengkap dan nomor hp. Sedangkan di gambar 4.2 terlihat halaman *log in* untuk pengguna yang telah melakukan registrasi. Untuk melakukan *log in* dibutuhkan alamat email dan password yang pernah didaftarkan sebelumnya.



Gambar 4.1 Tampilan Antarmuka Android Halaman *Register*



Gambar 4.2 Tampilan Antarmuka Android Halaman *Log In*



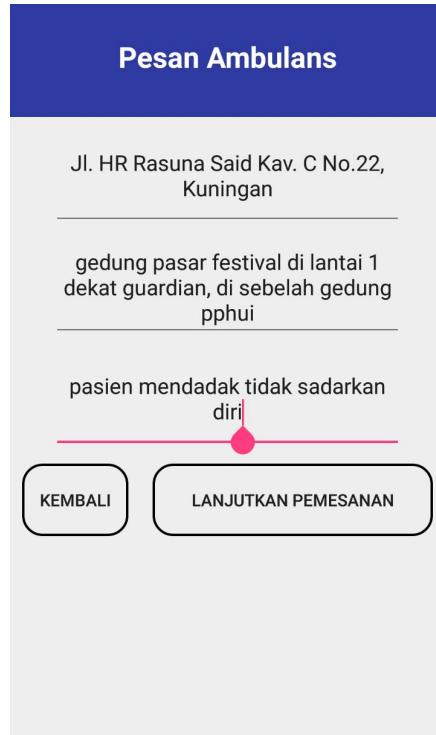
Gambar 4.3 Tampilan Antarmuka Android Halaman Menu

Bila pengguna telah melakukan pendaftaran dan *log in* pada aplikasi, maka akan terlihat tampilan menu utama aplikasi seperti pada gambar 4.3. Seperti yang dapat kita lihat, tampilan antarmuka untuk halaman menu pada gambar 4.3 terdapat beberapa tombol / *button* yaitu pesan ambulans, akun saya, dan *log out*. Tombol akun saya adalah fungsi untuk melihat dan mengubah (*edit*) beberapa informasi awal akun pengguna seperti nama, nomor hp, *email* dan *password*. Untuk melakukan pemesanan ambulans pada aplikasi OAM, pengguna yang telah *log in* dapat langsung melakukan klik pada tombol pesan ambulans. Sementara tombol *Log Out* digunakan ketika pengguna (*user*) ingin keluar dari akun aplikasi OAM. Halaman selanjutnya akan muncul seperti yang terlihat pada gambar 4.4, pengguna akan dimintai untuk memasukkan beberapa informasi dasar seperti alamat, patokan alamat, dan keterangan pasien yang akan dibawa menggunakan ambulans.

Hal ini dapat membantu mempersingkat waktu pemesanan karena informasi sudah detail di awal pemesanan. Untuk contoh pada saat formulir halaman pemesanan ambulans yang sudah di isi pada aplikasi dapat dilihat pada gambar 4.5. Informasi yang diisi sebagai contoh pengisian pada kolom alamat adalah Jl. HR Rasuna Said Kav. C No.22, Kuningan. Informasi yang diisi pada kolom patokan sebagai contoh pengisian adalah gedung pasar festival lantai 1 dekat guardian, disebelah gedung pphui. Lalu kolom deskripsi penyakit pada kasus ini akan dicontohkan keadaan pasien yang mendadak pingsan, sehingga diisi informasi di aplikasi bahwa pasien mendadak tidak sadarkan diri.



Gambar 4.4 Tampilan Antarmuka Android Halaman Pesan Ambulans



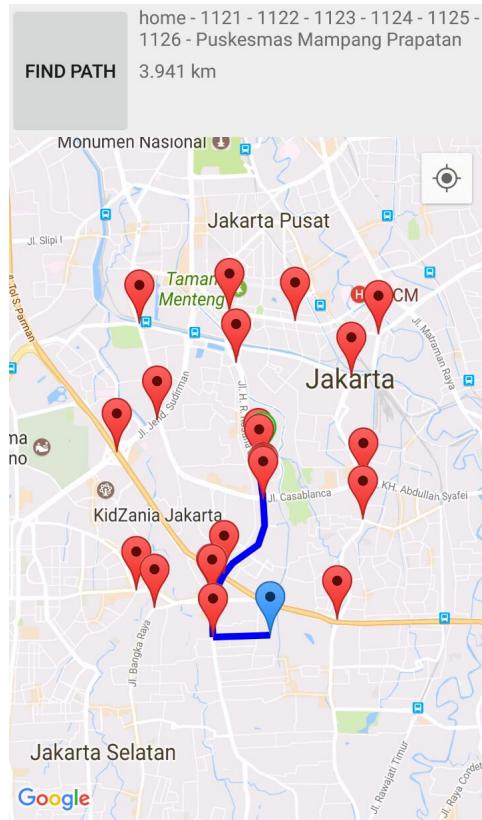
**Gambar 4.5 Tampilan Antarmuka Android Halaman Pesan Ambulans
(filled)**

Dalam mengisi formulir informasi pemesanan ambulans, aplikasi akan meminta pengguna untuk mengaktifkan GPS terlebih dahulu agar aplikasi dapat mendeteksi titik posisi pengguna dengan baik. Bila pengguna aplikasi tidak mengaktifkan / menolak pengaktifan sinyal GPS dan / atau tidak mengaktifkan paket internet pada *smartphoneny*a, maka aplikasi pemesanan ambulans OAM tidak dapat dijalankan karena kedua hal tersebut dibutuhkan untuk memesan ambulans melalui aplikasi OAM. Gambar 4.6 Merupakan halaman yang akan ditampilkan ketika pengguna menekan tombol “Lanjutkan Pemesanan” pada gambar 4.6 sebelumnya. Halaman yang ada pada gambar 4.6 menunjukkan titik – titik rute (nodes) , titik lokasi pemesan, dan titik pos yang tersedia dalam radius 3 km.



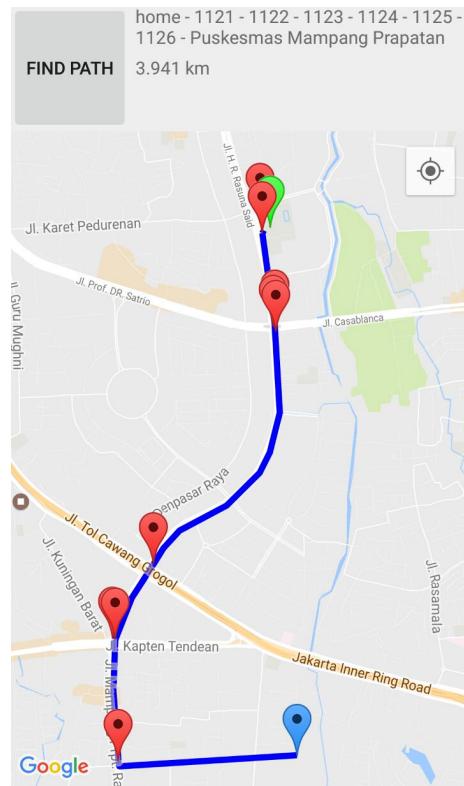
Gambar 4.6 Tampilan Antarmuka Android Halaman Pesan Ambulans dengan Algoritma Dijkstra

Titik berwarna merah merupakan titik rute yang akan dilalui oleh mobil ambulans dalam menjemput pemesan(*user*) dari pos terdekat. Titik berwarna biru merupakan titik pos yang tersedia disekitar pemesan (*user*). Sedangkan titik berwarna hijau merupakan titik lokasi pemesan ambulans (*user*). Dalam satu daerah bisa saja mengandung 2 atau lebih pos ambulans yang tersedia dalam radius yang telah ditentukan Pada gambar 4.6, setelah pengguna menekan tombol / button “find path” maka akan muncul tampilan aplikasi seperti 4.7. Dalam proses pemesanan ini, sudah terjadi pengiriman informasi dari aplikasi OAM Android ke *web service* OAM.



Gambar 4.7 Tampilan Antarmuka Android Halaman Pesan Ambulans dengan Algoritma Dijkstra Dua

Admin *website* OAM akan menerima informasi data pemesanan ambulans berupa nama, nomor hp, alamat, patokan alamat, dan deskripsi pasien. Apabila pengguna menekan tombol “*find path*” kembali pada gambar 4.7, maka aplikasi akan segera memunculkan tampilan hasil aplikasi seperti pada gambar 4.8. Dalam gambar 4.8 sudah terjadi pengiriman informasi data berupa status, status *handled* bila pemesan masih berada di Jakarta atau status *out of coverage* bila pemesan berada di luar Jakarta dan pos terdekat. Pada gambar 4.8 terlihat rute terdekat yang ada sebagai hasil algoritma di dalam aplikasi. 1121, 1122, 1123, 1124, 1125, 1126 merupakan nama titik-titik rute yang telah terdaftar pada aplikasi OAM.



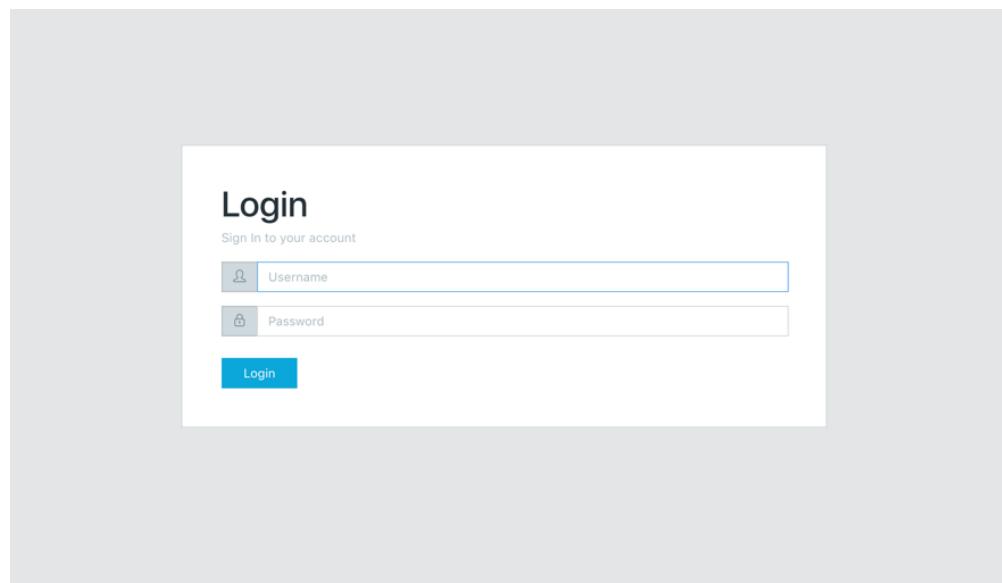
Gambar 4.8 Tampilan Antarmuka Android Halaman Pesan Ambulans dengan Algoritma Dijkstra Tiga

Pop up box bertuliskan “Anda telah berhasil memesan ambulans, Silahkan tunggu konfirmasi dari admin OAM” akan muncul ketika proses pemesanan ambulans melalui aplikasi telah selesai dan berhasil dilakukan. Hal itu menjadi patokan keberhasilan dalam memesan ambulans menggunakan aplikasi OAM. Selanjutnya pihak admin ambulans akan menghubungi pengguna (*user*) aplikasi melalui nomor telepon yang diberikan. Bila pemesanan tidak dapat di konfirmasi, admin ambulans akan mengubah status pemesanan menjadi *declined* dan tidak akan menindak lanjuti pemesanan ambulans tersebut. Apabila pemesanan dapat dikonfirmasi kebenarannya, admin ambulans akan

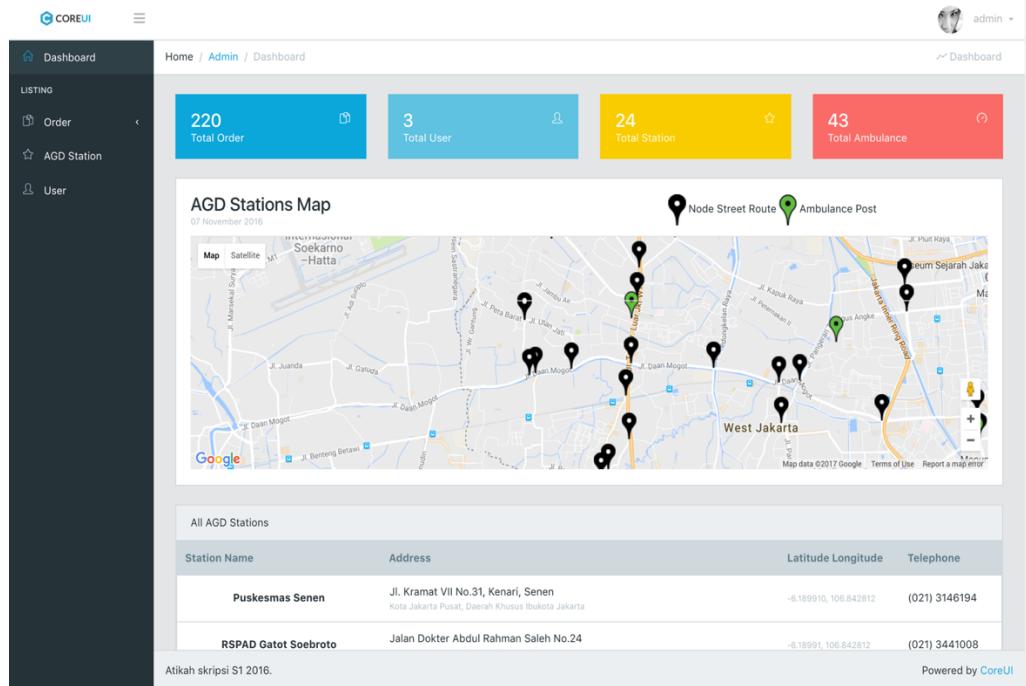
langsung menindaklanjuti pemesanan ambulans dengan cara mengirimkan ambulans ke posisi tujuan.

4.4.3 Implementasi Aplikasi OAM pada *Website Admin*

Implementasi OAM pada *website admin* dilakukan berdasarkan perencanaan awal yang telah dilakukan. Perancangan aplikasi OAM sudah dibahas secara lengkap di bab II pada fase desain aplikasi dari hasil identifikasi sebelumnya. Untuk *website OAM* telah digunakan template *Core UI* yang bersifat *open source*. Template ini memiliki ketergantungan pada *Bootstrap 3*, *JQuery 1.11+*, serta *plugin* lainnya agar bisa menampilkan elemen antarmuka dengan baik. Halaman antarmuka website OAM untuk *log in* admin dapat dilihat pada gambar 4.9. Masing – masing admin akan disediakan *username* dan *passwordnya* untuk mengakses halaman manajemen pemesanan ambulans di OAM. Setelah melakukan *log in* dengan cara memasukkan *username* dan *password* yang tepat, akan muncul tampilan halaman menu utama website seperti pada gambar 4.10.

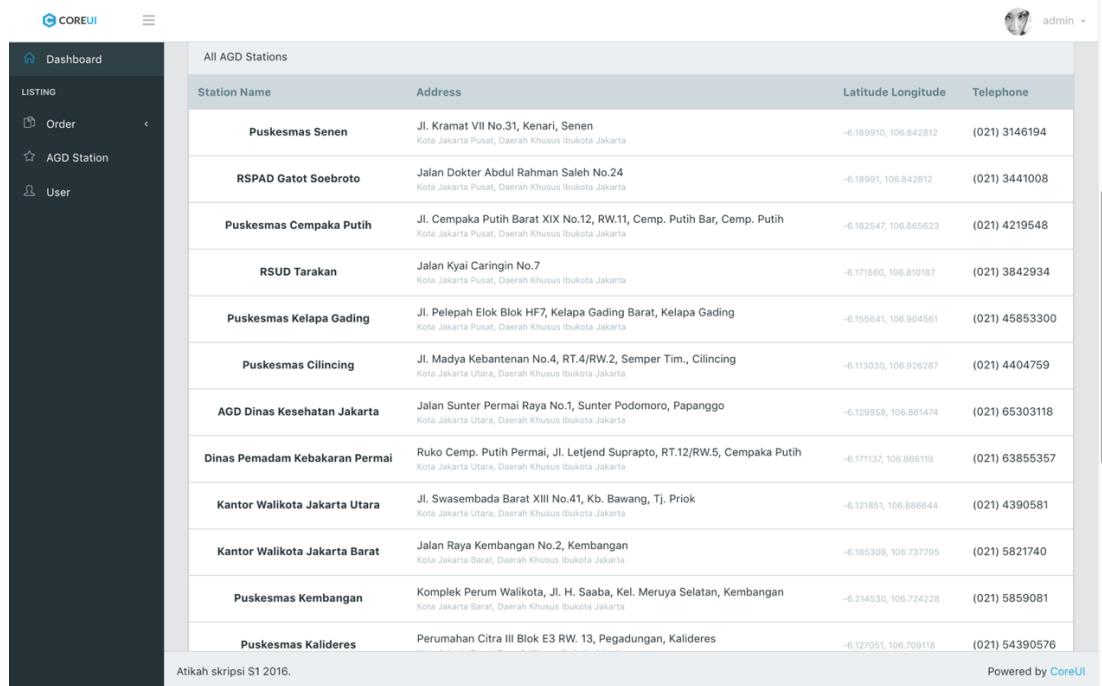


Gambar 4.9 Tampilan Antarmuka *Website OAM* Halaman *Log In* Admin



Gambar 4.10 Tampilan Antarmuka Website OAM Halaman Main Menu

Pada halaman *main menu website* OAM di gambar 4.10, terlihat beberapa informasi yang langsung ditampilkan. Admin ambulans dapat melihat informasi jumlah seluruh pemesanan, pengguna (*user*), pos ambulans (*station*), dan jumlah ambulans di halaman utama. Selanjutnya terlihat informasi titik – titik rute (*node street route*) dan titik-titik pos ambulans secara langsung di *Google Map*. Bila di scroll lebih lanjut kebawah, akan terlihat informasi pos – pos ambulans yang ada pada AGD 118 di Jakarta. Tampilan nformasi pos – pos ambulans dapat dilihat pada gambar 4.11.



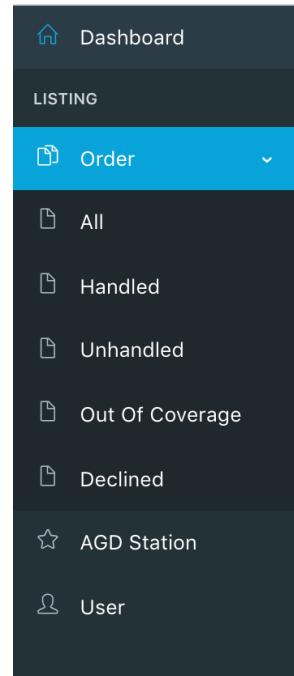
The screenshot shows a web application interface for managing Ambulans Gudang Daerah (AGD) stations. The top navigation bar includes the COREUI logo and a user profile for 'admin'. On the left, a sidebar menu lists 'Dashboard', 'LISTING' (with 'Order', 'AGD Station', and 'User' options), and a 'Powered by CoreUI' footer. The main content area is titled 'All AGD Stations' and displays a table with columns: 'Station Name', 'Address', 'Latitude Longitude', and 'Telephone'. The table lists ten stations, each with its name, address, coordinates, and phone number.

Station Name	Address	Latitude Longitude	Telephone
Puskesmas Senen	Jl. Kramat VII No.31, Kenari, Senen Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.189910, 106.842812	(021) 3146194
RSPAD Gatot Soebroto	Jalan Dokter Abdul Rahman Saleh No.24 Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.18991, 106.842812	(021) 3441008
Puskesmas Cempaka Putih	Jl. Cempaka Putih Barat XIX No.12, RW.11, Cemp. Putih Bar, Cemp. Putih Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.182547, 106.865623	(021) 4219548
RSUD Tarakan	Jalan Kyai Caringin No.7 Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.171560, 106.810187	(021) 3842934
Puskesmas Kelapa Gading	Jl. Pelepas Elo Blok HF7, Kelapa Gading Barat, Kelapa Gading Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.155841, 106.904561	(021) 45853300
Puskesmas Cilincing	Jl. Madya Kebantenan No.4, RT.4/RW.2, Semper Tim., Cilincing Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta	-6.113030, 106.926287	(021) 4404759
AGD Dinas Kesehatan Jakarta	Jalan Sunter Permai Raya No.1, Sunter Podomoro, Papanggo Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta	-6.129858, 106.861474	(021) 65303118
Dinas Pemadam Kebakaran Permai	Ruko Cemp. Putih Permai, Jl. Letjend Suprapto, RT.12/RW.5, Cempaka Putih Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta	-6.171137, 106.866119	(021) 63855357
Kantor Walikota Jakarta Utara	Jl. Swasembada Barat XIII No.41, Kb. Bangow, Tj. Priok Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta	-6.121851, 106.886644	(021) 4390581
Kantor Walikota Jakarta Barat	Jalan Raya Kembaran No.2, Kembaran Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta	-6.185308, 106.737795	(021) 5821740
Puskesmas Kembangan	Komplek Perum Walikota, Jl. H. Saaba, Kel. Meruya Selatan, Kembangan Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta	-6.214530, 106.724228	(021) 5859081
Puskesmas Kalideres	Perumahan Citra III Blok E3 RW. 13, Pegadungan, Kalideres	-6.127051, 106.709118	(021) 54390576

Atikah skripsi S1 2016. Powered by CoreUI

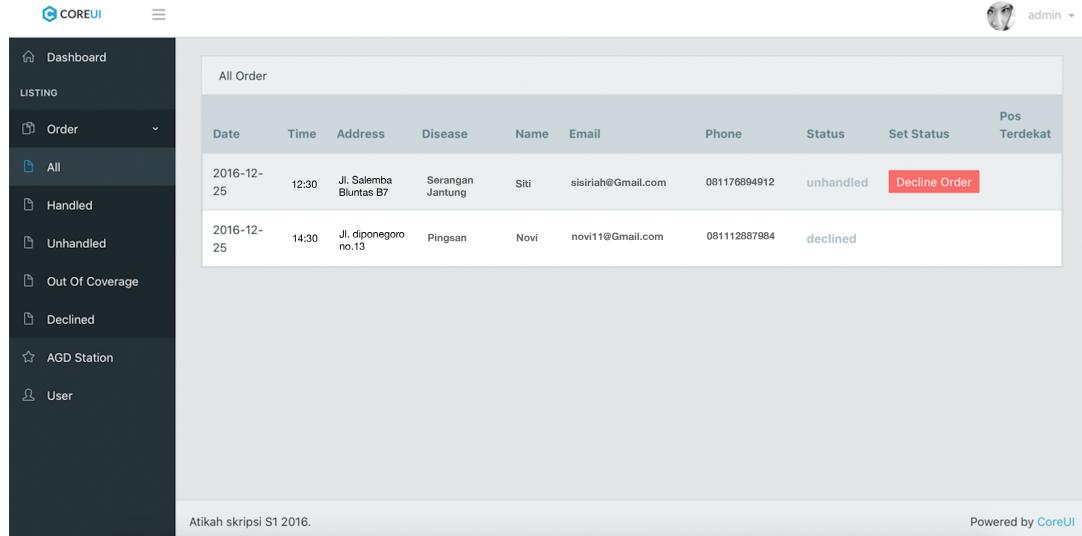
Gambar 4.11 Tampilan Antarmuka Website OAM Halaman Main Menu Dua

Informasi pos – pos ambulans yang ditampilkan pada website admin berupa informasi seperti nama pos (*station name*), alamat pos (*address*), titik *longitude* dan titik *latitude* (*longitude latitude*), serta nomor teleponnya (*Telephone*). Pada sidebar menu di sebelah kiri terdapat tombol *Order*, *AGD Stations*, dan *Users*. Apabila tombol Order di klik, maka akan muncul beberapa tambahan menu yang terletak dibawahnya. Tambahan – tambahan menu yang muncul berupa *All*, *Handled*, *Unhandled*, *Out Of Coverage*, dan *Declined* seperti yang terlihat pada gambar 4.12. Tombol – Tombol tersebut merupakan jenis status yang terdapat pada proses pemesanan ambulans.



Gambar 4.12 Tampilan Antarmuka Website OAM Menu Sidebar

Informasi pemesanan ambulans dapat dilihat secara menyeluruh melalui tombol *All* ataupun secara spesifik misalnya hanya ingin melihat informasi pemesanan ambulans yang berstatus *Out Of Coverage* dapat langsung menekan tombol *Out Of Coverage*. Salah satu contoh tampilan halaman pemesanan ambulans untuk status *Handled* dapat dilihat pada gambar 4.13. Setiap halaman yang memiliki informasi pemesanan ambulans yang sama yaitu nama pemesan, email, nomor telepon, tanggal, waktu, alamat, status, tombol mengubah status menjadi *declined*, dan nama pos terdekat dari pengguna (*user*). Tampilan halaman untuk sidebar menu “AGD Stations” dapat dilihat pada gambar 4.14. Pada halaman tersebut akan terlihat informasi pos – pos ambulans yang ada pada AGD 118 di Jakarta.



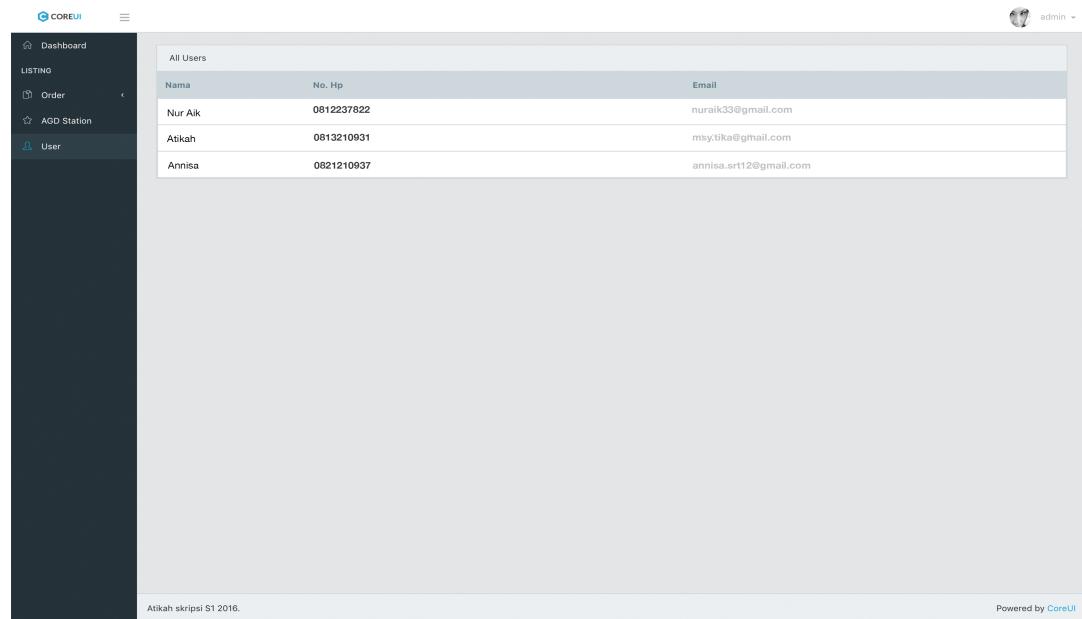
Gambar 4.13 Tampilan Antarmuka Website OAM Halaman Pemesanan Ambulans Berstatus *Handled*

All AGD Stations				
Station Name	Address	Latitude	Longitude	Telephone
Puskesmas Senen	Jl. Kramat VII No.31, Kenari, Senen Kota Jakarta Pusat, Daerah Khusus Ibu Kota Jakarta	-6.189910	106.842812	(021) 3140194
RSPAD Gatot Soebroto	Jalan Dokter Abdul Rahman Saleh No.24 Kota Jakarta Pusat, Daerah Khusus Ibu Kota Jakarta	-6.18991	106.842812	(021) 3441008
Puskesmas Cempaka Putih	Jl. Cempaka Putih Barat XIX No.12, RW.11, Cemp. Putih Bar, Cemp. Putih Kota Jakarta Pusat, Daerah Khusus Ibu Kota Jakarta	-6.182547	106.865423	(021) 4219548
RSUD Tarakan	Jalan Kyai Caringin No.7 Kota Jakarta Pusat, Daerah Khusus Ibu Kota Jakarta	-6.171580	106.810187	(021) 3842934
Puskesmas Kelapa Gading	Jl. Pelepas Blok HF7, Kelapa Gading Barat, Kelapa Gading Kota Jakarta Pusat, Daerah Khusus Ibu Kota Jakarta	-6.156641	106.904561	(021) 45853300
Puskesmas Cilincing	Jl. Madya Kebantenan No.4, RT.4/RW.2, Semper Tim., Cilincing Kota Jakarta Utara, Daerah Khusus Ibu Kota Jakarta	-6.113030	106.926287	(021) 4404759
AGD Dinas Kesehatan Jakarta	Jalan Sunter Permai Raya No.1, Sunter Podomoro, Papangga Kota Jakarta Utara, Daerah Khusus Ibu Kota Jakarta	-6.129888	106.861474	(021) 65303118
Dinas Pemadam Kebakaran Permai	Ruko Cemp. Putih Permai, Jl. Letjend Suprapto, RT.12/RW.5, Cempaka Putih Kota Jakarta Utara, Daerah Khusus Ibu Kota Jakarta	-6.771137	106.866119	(021) 63855357
Kantor Walikota Jakarta Utara	Jl. Swasembada Barat XIII No.41, Kb. Bawang, Tj. Priok Kota Jakarta Utara, Daerah Khusus Ibu Kota Jakarta	-6.121851	106.886644	(021) 4390581
Kantor Walikota Jakarta Barat	Jalan Raya Kembaran No.2, Kembaran Kota Jakarta Barat, Daerah Khusus Ibu Kota Jakarta	-6.185308	106.737795	(021) 5821740
Puskesmas Kembangan	Kompleks Perum Walikota, Jl. H. Saabu, Kel. Meruya Selatan, Kembangan Kota Jakarta Barat, Daerah Khusus Ibu Kota Jakarta	-6.214830	106.724228	(021) 5859081
Puskesmas Kalideres	Perumahan Citra III Blok E3 RW. 13, Pegadungan, Kalideres Kota Jakarta Barat, Daerah Khusus Ibu Kota Jakarta	-6.127051	106.709118	(021) 54390576
Puskesmas Grogol Petamburan	Kompleks Tamans Duta May, Jl. Wijaya Kusuma 3 Blok. F No. 1 Kota Jakarta Barat, Daerah Khusus Ibu Kota Jakarta	-6.140681	106.776740	(021) 5648379
Puskesmas Cendakromo	Jalan Kamal Raya No. 02, Cengkareng Bar., Cengkareng Kota Jakarta Barat, Daerah Khusus Ibu Kota Jakarta	-6.144374	106.728240	(021) 29038167

Gambar 4.14 Tampilan Antarmuka Website OAM Halaman AGD Stations

Informasi yang ditampilkan sama dengan yang ditampilkan pada menu utama website di gambar 4.11 yaitu nama pos, alamat, titik longitude dan titik latitude, dan nomor telepon tiap – tiap pos. Tampilan halaman untuk *sidebar menu*

“Users” dapat dilihat pada gambar 4.15. Pada halaman tersebut akan terlihat informasi pengguna aplikasi yang melakukan pendaftaran dan telah melakukan pemesanan ambulans. Informasi pengguna aplikasi (*user*) yang tertera pada *website* berupa nama, nomor hp, dan alamat email.



The screenshot shows a web application interface for managing users. On the left, there is a dark sidebar with navigation links: Dashboard, LISTING, Order, AGD Station, and User. The 'User' link is highlighted in blue. The main content area has a header 'All Users'. Below it is a table with three columns: 'Nama', 'No. Hp', and 'Email'. The table contains three rows of data:

Nama	No. Hp	Email
Nur Aik	0812237822	nuraik33@gmail.com
Atikah	0813210931	msy.likka@gmail.com
Annisa	0821210937	annisa.art12@gmail.com

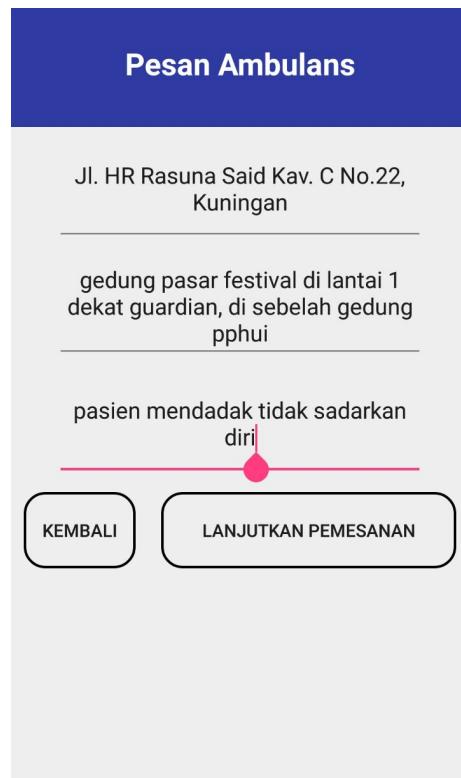
At the bottom of the page, there is a footer note 'Atikah skripsi S1 2016.' and a 'Powered by CoreUI' link.

Gambar 4.15 Tampilan Antarmuka Website OAM Users

4.2 Pengujian Aplikasi OAM

Tahap ini merupakan tahap pengujian (*testing*) karena pengujian aplikasi OAM akan dilakukan pada bab ini. Seperti yang dibahas pada BAB II, Algoritma Dijkstra yang digunakan pada aplikasi OAM merupakan algoritma pencarian jarak terpendek (*shortest path algorithm*) merupakan solusi untuk mencari jarak terpendek pada lintasan atau rute dari titik awal hingga titik tujuan. Algoritma Dijkstra ditemukan oleh Edsger Dijkstra. Algoritma ini merupakan algoritma yang paling sering digunakan dalam pencarian rute terpendek. Penggunaannya dengan menggunakan simpul simpul sederhana pada jaringan jalan yang tidak rumit menjadikannya algoritma yang sederhana untuk digunakan (Chamero, 2006). Untuk menguji aplikasi maka akan

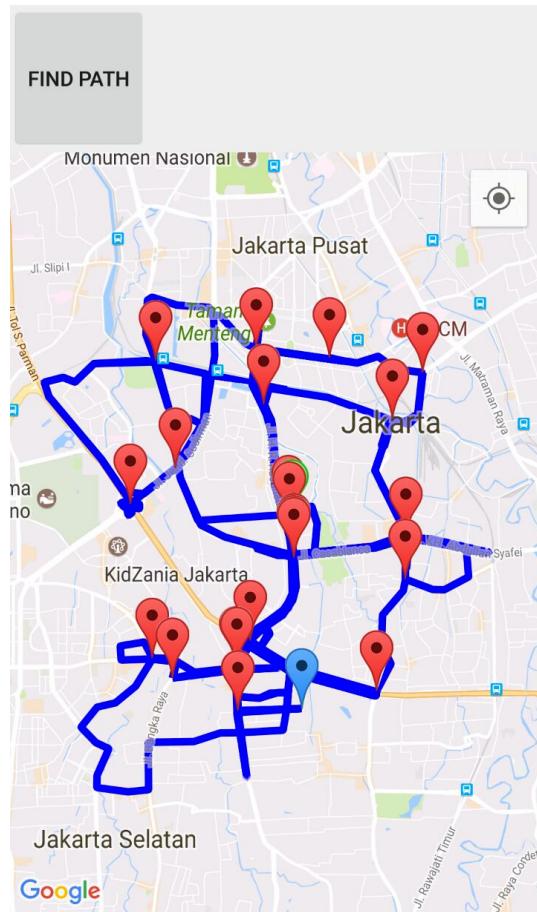
dibuat sebuah kondisi pemesanan sebagai contoh kasus untuk menjalankan aplikasi pemesanan ambulans OAM. Lokasi pengguna (*user*) yang ditentukan untuk melakukan pemesanan ambulans melalui aplikasi akan dicontohkan pada jalan Haji R. Rasuna Said Kav C No.22 Jakarta Selatan, dengan titik *longitude* -6.221336 dan titik *latitude* nya 106.832882.



Gambar 4.16 Tampilan Antarmuka Android Halaman Pesan Ambulans (*filled*)

Pengguna (*user*) yang telah melakukan pendaftaran dan *log in* aplikasi, akan memasukkan beberapa informasi untuk pemesanan seperti alamat, patokan alamat, dan deskripsi penyakit pasien. Setelah mengisi informasi tersebut, pengguna (*user*) akan melanjutkan proses pemesanan dengan klik tombol “lanjutkan pemesanan” seperti yang terlihat pada gambar 4.16. Aplikasi akan merespon tindakan tersebut dengan

menampilkan halaman peta dan pencarian pos terdekat. Tampilan selanjutnya terlihat pada gambar 4.17. Pada contoh kasus pemesanan ini terdapat total *nodes* berjumlah 21 titik yang terdiri dari 1 titik lokasi pemesan, 19 node rute, dan terdeteksi 1 pos ambulans dalam radius 3 km.



Gambar 4.17 Tampilan Antarmuka Android Halaman Pesan Ambulans

Pada saat pengguna (*user*) klik “lanjutkan pemesanan” pada halaman pengisian informasi pemesanan di gambar 4.16, aplikasi akan mengirimkan data laporan informasi ke *web service*. Lalu *web service* akan memberikan informasi titik-titik rute (*nodes*) dan pos – pos ambulans terdekat yang tersedia dalam 3 km. Kodingan aplikasi yang ditampilkan pada Android Studio terlihat pada gambar 4.18.

```

PesananAmbulans.create({
    "alamat_masyarakat": alamat, "patokan_alamat": patokan, "keterangan": deskripsi,
    "latitude": latitude, "longitude": longitude, "masyarakatId": uid, "status": STATUS_UNHANDLED,
    "tanggal": moment().tz("Asia/Jakarta").format("YYYY-MM-DD"), "waktu": moment().tz('Asia/Jakarta').format("HH:mm:ss")
}, function (err, lapor) {
    if (err) {
        return callback(null, "error " + err);
    }

    PesananAmbulans.getApp(function (err, app) {
        app.models.PosAmbulans.find({
            where: {
                posisi: {near: userLocation, maxDistance: 3, unit: 'kilometers'}
            }
        }, function (err, pos) {

            if (err || pos == null || pos.length < 1){
                return callback(null, {pos: [], pesan: lapor});
            }

            var homeBase = {
                "nama": "home",
                "posisi": {"lat": latitude, "lng": longitude},
                "is_pos": false,
                "vertex": [{"awal": "home", "akhir": pos[0]['nama']}]
            };
            console.log("test data " + homeBase + " " + pos.length);

            pos.push(homeBase);
            console.log("test data1 " + " " + pos.length);

            var data = {
                pos: pos,
                pesan: lapor
            };

            return callback(null, data);
        });
    });
});

```

Gambar 4.18 Sebagian Kodingan pada Android Studio

Kodingan pada gambar 4.18 tersebut menjelaskan bahwa aplikasi hanya akan menampilkan pos yang berada dalam radius 3 km di sekitar pemesan (*user*) ambulans, hal tersebut didefiniskan pada garis ini “app.models.PosAmbulans.find ({where: {posisi: {near:userLocation,maxDistance: 3, unit: 'kilometers'}}}”. Kodingan untuk inisialisasi titik-titik rute jalan (*nodes*) dapat dilihat pada gambar 4.19.

```

public Dijkstra(JSONArray jsonArrayNode){
    this.jsonArrayNode = jsonArrayNode;
    initiateNode();
}

```

Gambar 4.19 Sebagian Kodingan pada Android Studio Dua

InitiateNode merupakan *method* dalam algoritma dijkstra untuk proses inisialisasi titik-titik rute jalan (*nodes*). Dalam *method* tersebut setiap titik rute memiliki tetangga titik rute yang bersebelahan / satu lintas dan titik posisi berupa titik longitude dan latitudenya. Penulisan kodingan method *initiateNode* dapat dilihat pada gambar 4.20

```

private void initiateNode(){
    for (int i=0;i<jsonArrayNode.length();i++){
        Node node = new Node();
        String strNama = jsonArrayNode.optJSONObject(i).optString("nama");
        JSONArray jsonArrayVertex = jsonArrayNode.optJSONObject(i).optJSONArray("vertex");
        String strPosition = jsonArrayNode.optJSONObject(i).optString("position");
        node.addNode(strNama,jsonArrayVertex,strPosition);
        arrayListNode.add(node);
        try {
            jsonObjectNodePosition.put(strNama,i);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}

```

Gambar 4.20 Sebagian Kodingan pada Android Studio Tiga

Algoritma Dijkstra pada aplikasi OAM secara keseluruhan digambarkan pada kodingan Android yang terlihat pada gambar 4.21. Method *execute* ini merupakan inti dari algoritma dijkstra yang didalamnya terbagi menjadi dua proses utama untuk dijalankan. Proses yang pertama adalah proses *looping* untuk mencari rute terdekat pada tiap-tiap titik rute (*nodes*). Hasil dari proses pertama masih dalam keadaan terbalik. Misalnya rute yang dihasilkan adalah D,C,B,A padahal seharusnya tertulis A,B,C,D.

```

public JSONObject execute(String strAwal, String strAkhir){
    strActiveName = strAwal;
    strActiveAkhir = strAkhir;
    endPoint = jsonObjectNodePosition.optInt(strAkhir);

    for (int i = 0; i < arrayListNode.size(); i++) {
        if (arrayListNode.get(endPoint).isActive()){
            traceNode();
            totalWeight = initialWeight;
        } else {
            break;
        }
    }

    traceBack(strAwal, strAkhir);
    JSONArray jsonArrayTemp = new JSONArray();
    for (int i = jsonArrayPath.length() - 1; i >= 0; i--) {
        jsonArrayTemp.put(jsonArrayPath.optString(i));
        if (i == 0){
            jsonArrayPath = jsonArrayTemp;
        }
    }

    JSONObject jsonObjectResult = new JSONObject();
    try {
        jsonObjectResult.put("path", jsonArrayPath);
        jsonObjectResult.put("distance", totalWeight);
    } catch (JSONException e) {
        e.printStackTrace();
    }

    return jsonObjectResult;
}

```

Gambar 4.21 Sebagian Kodingan pada Android Studio Empat

Sehingga dibutuhkan proses membalikkan hasil yang ditemukan melalui proses kedua yaitu *traceback*. Hasil dari *traceback* ini merupakan hasil akhir dari jalannya algoritma Dijkstra pada aplikasi OAM yang juga ditampilkan pada aplikasi. Selain ditampilkan pada aplikasi, informasi pos terdekat juga akan dikirimkan langsung ke *web service* untuk diketahui oleh admin ambulans. Kode penulisan method *traceNode* dapat dilihat pada gambar 4.22. Dalam melakukan *traceNode*, akan didapatkan nilai jarak terdekat

dari titik/node tetangga. Pada method traceNode, perlu dipastikan bahwa titik *node* aktif yang sedang dilakukan *trace* bukan merupakan target akhir dari proses algoritma. Bila bukan target titik akhir, maka nilai intBestWeightLinear akan di set menjadi nol. Lalu akan dicari node aktif menggunakan method findNode(String strActiveName); .

```

private void traceNode() {
    strBestNameLinear = "";
    if (!strActiveName.equalsIgnoreCase(strActiveAkhir)) {
        intBestWeightLinear = 0;
    }
    activeNode = findNode(strActiveName);
    arrayListTetangga = activeNode.getArrayListTetangga();

    for (int i=0;i<activeNode.getVertexLength();i++){
        String strTetanggaName = arrayListTetangga.get(i).getStrAkhir();
        int intTetanggaWeight = arrayListTetangga.get(i).getIntWeight()+initialWeight;
        int intTetanggaPoint = jsonObjectNodePosition.optInt(strTetanggaName);
        Node tetanggaNode = arrayListNode.get(intTetanggaPoint);
        tetanggaNode.setBestTetangga(intTetanggaWeight,activeNode.getNodeName());

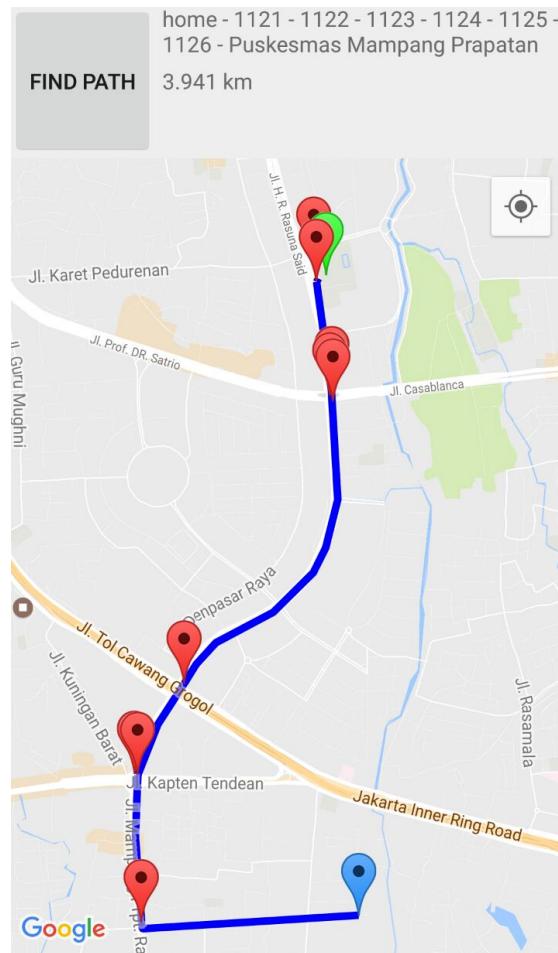
        if (i == activeNode.getVertexLength() - 1) {
            activeNode.setActive(false);
            getBestValue();
            strActiveName = strBestNameLinear;
            initialWeight = intBestWeightLinear;
        }
    }
}

```

Gambar 4.22 Sebagian Kodingan pada Android Studio Lima

Kemudian dari node aktif tersebut di ambil / *collect* list-list tetangga yang ada pada node aktif tersebut. Dari list tetangga pada node aktif tersebut akan dilakukan *looping* sehingga pada node tetangga dari node yang aktif tersebut akan di set jarak dan node terbaiknya. Pada akhir *looping* pada tiap tetangga, node yang aktif tersebut akan di set menjadi tidak aktif dan dicari titik terendah jarak terdekat/terbaik berikutnya untuk melanjutkan *traceNode*. Contoh hasil akhir dari algoritma Dijkstra yang ditampilkan pada aplikasi OAM dapat dilihat pada gambar 4.23. Aplikasi menunjukkan titik posisi awal pemesan dan rute dari pos ambulans terdekat. Nama titik-titik rute yang dilalui

adalah 1121, 1122, 1123, 1124, 1125, dan 1126. Nama pos yang terdekat dari pemesan ambulans adalah Puskesmas Mampang Prapatan yang berjarak 3,9km dari titik lokasi pemesan (*user*).



Gambar 4.23 Tampilan Antarmuka Android Pesan Ambulans

4.2.1 *White Box Testing*

Salah satu metode yang dilakukan untuk mencari tahu keberadaan error dalam aplikasi dengan mempertimbangkan mekanisme logika sistem adalah *white box testing* (Khan, 2011). *White Box* adalah metode untuk menguji suatu aplikasi atau *software* dengan cara melihat modul untuk dapat meneliti dan menganalisa kode dari program yang di buat ada yang salah atau tidak. Apabila modul menghasilkan *output* yang tidak

sesuai dengan ekspektasi, maka akan dilakukan perbaikan terhadap kode-kode tersebut (Nidhra, 2012). Kasus yang sering menggunakan white box testing akan di uji dengan beberapa tahapan yaitu:

1. Pengujian seluruh keputusan yang menggunakan logikal.
2. Pengujian keseluruhan loop yang ada sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

Pengujian menggunakan *white box* diterapkan pada beberapa fungsi Algoritma Dijkstra. Hasil dari pengujian *white box* terhadap sebagian fungsi penting Algoritma Dijkstra pada aplikasi OAM dapat dilihat di sebagai berikut :

```
private void initiateNode(){
    for (int i=0;i<jsonArrayNode.length();i++){
        Node node = new Node();
        String strNama = jsonArrayNode.optJSONObject(i).optString("nama");
        JSONArray jsonArrayVertex = jsonArrayNode.optJSONObject(i).optJSONArray("vertex");
        String strPosition = jsonArrayNode.optJSONObject(i).optString("position");
        node.addNode(strNama,jsonArrayVertex,strPosition);
        arrayListNode.add(node);
        try {
            jsonObjectNodePosition.put(strNama,i);
        } catch (JSONException e) {
            e.printStackTrace();
        }
    }
}
```

Gambar 4.24 Fungsi initiateNode()

Nama Fungsi	initiateNode()
Deskripsi	Fungsi ini digunakan untuk mengetahui seluruh titik – titik hasil respon dari <i>web service</i> . Lihat gambar 4.24 untuk fungsi initiateNode()
Input	Sebuah <i>json array</i> yang berisikan nama titik / <i>node</i> dengan tipe <i>string</i> , tetangga yang bertipe <i>json array</i> , dan posisi titik <i>longitude</i> dan <i>latitude</i> yang berupa <i>string</i> .

Output	Mengetahui letak posisi titik/node dalam variabel <i>arrayListNode</i> seperti yang terlihat pada kodingan (jsonObjectNodePosition.put(strNama, i)).
Status	Succeed

Tabel 4.1 Fungsi initiateNode()

```

public ArrayList<Vertex> getAllVertex(){
    //Log.i("test","starting to get all vertex");
    ArrayList<Vertex> arrayListVertex = new ArrayList<Vertex>();
    for (int i=0;i<arrayListNode.size();i++){
        for (int j=0;j<arrayListNode.get(i).getArrayListTetangga().size();j++){
            arrayListVertex.add(arrayListNode.get(i).getArrayListTetangga().get(j));
            Log.i("test", "getting all vertex round "+i+" node name "+arrayListNode.get(i).getNodeName()+
                  " start vertex round "+j+" vertex node awal name "+arrayListNode.get(i).getArrayListTetangga().get(j).getStrAwal()+
                  " and vertex node akhir "+arrayListNode.get(i).getArrayListTetangga().get(j).getStrAkhir());
        }
    }
    return arrayListVertex;
}

```

Gambar 4.25 Fungsi getAllVertex()

Nama Fungsi	getAllVertex()
Deskripsi	Fungsi ini digunakan untuk mengetahui titik tetangga dari seluruh titik yang diketahui dari hasil respon dari <i>web service</i> . Lihat gambar 4.25 untuk fungsi getAllVertex()
Input	<i>arrayListNode</i> , sebuah <i>array</i> dari <i>variabel node</i> yang didalamnya terdapat daftar (<i>lists</i>) tetangga dari tiap titik (<i>node</i>) untuk di panggil kembali.
Output	<i>arrayListVertex</i> , sebuah <i>array</i> yang bertipe <i>vertex</i> (tetangga) yang berisikan semua titik yang terhubung yang memiliki titik awal dan akhir.
Status	Succeed

Tabel 4.2 Fungsi getAllVertex()

```

private void sendRequest() {
    jsonArrayLocation = new JSONArray();
    try {
        for (int i = 0; i < arrayListVertex.size(); i++) {
            int intPositionOrigin = dijkstra.getNodePosition(arrayListVertex.get(i).getStrAwal());
            int intPositionDestination = dijkstra.getNodePosition(arrayListVertex.get(i).getStrAkhir());

            JSONObject jsonObjectLocation = new JSONObject();
            jsonObjectLocation.put("origin", jsonArrayNodeFix.optJSONObject(intPositionOrigin).optJSONObject("posisi").optString("lat")
                + "," + jsonArrayNodeFix.optJSONObject(intPositionOrigin).optJSONObject("posisi").optString("lng"));
            jsonObjectLocation.put("destination", jsonArrayNodeFix.optJSONObject(intPositionDestination).optJSONObject("posisi").optString("lat")
                + "," + jsonArrayNodeFix.optJSONObject(intPositionDestination).optJSONObject("posisi").optString("lng"));
        };
        jsonArrayLocation.put(jsonObjectLocation);
    }

    for (int i = 0; i < jsonArrayNodeFix.length(); i++) {
        String strLocation = jsonArrayNodeFix.optJSONObject(i).optJSONObject("posisi").optString("lat") + "," +
            jsonArrayNodeFix.optJSONObject(i).optJSONObject("posisi").optString("lng");
        Double doubleLatitude = Double.valueOf(strLocation.split(",")[0]);
        Double doubleLongitude = Double.valueOf(strLocation.split(",")[1]);
        LatLng latLng = new LatLng(doubleLatitude, doubleLongitude);

        mMap.addMarker(new MarkerOptions()
            .icon(getMarkerFlat(jsonArrayNodeFix.optJSONObject(i).optBoolean("is_pos"), i))
            .title(jsonArrayNodeFix.optJSONObject(i).optString("nama"))
            .position(latLng));
    }
}

} catch (JSONException e) {
    e.getMessage();
}
}

```

Gambar 4.26 Fungsi sendRequest()

Nama Fungsi	sendRequest()
Deskripsi	Fungsi ini digunakan untuk mengirimkan data ke <i>web service</i> untuk mendapatkan titik posisi pemesan (<i>user</i>) dan pos – pos ambulans terdekat dalam radius 3 km dari titik posisi pengguna (<i>user</i>).
Input	<i>jsonArrayNodeFix</i> , sebuah <i>json array</i> dari variabel <i>Node</i> yang didalamnya terdapat daftar (<i>lists</i>) titik (<i>node</i>) berada dalam radius 3 km yang merupakan hasil respon dari <i>web service</i> . Seperti yang terlihat pada Gambar 4.26 yang menunjukkan fungsi sendRequest()
Output	Titik awal (<i>strHome</i>) adalah titik (<i>node</i>) terletak pada <i>index</i> terakhir dari <i>jsonArrayNodeFix</i> dan bukan merupakan pos (<i>boolean is_pos = false</i>). Titik akhir (<i>arrayListPos</i>) adalah titik (<i>node</i>) yang bertipe <i>array</i> karena memiliki kemungkinan titik yang lebih dari satu dan memiliki kondisi bahwa titik merupakan pos (<i>boolean is_pos = true</i>) pada <i>jsonArrayNodeFix</i> . Lihat kodingan pada gambar Gambar 4.27 yang menunjukkan fungsi <i>getMarkerFlat(boolean is_pos, int intPosition)</i> .
Status	Succeed

Tabel 4.3 Fungsi sendRequest()

```

private BitmapDescriptor getMarkerFlat(boolean is_pos, int intPosition) {
    if (is_pos) {
        arrayListPos.add(jsonArrayNodeFix.optJSONObject(intPosition).optString("nama"));
        arrayListDijkstra.add(new Dijkstra(jsonArrayNodeFix));
        Log.i("test", "testgggs" + jsonArrayNodeFix.optJSONObject(intPosition).optString("nama"));
        return BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_AZURE);
    } else if (!is_pos && intPosition == jsonArrayNodeFix.length() - 1) {
        Log.i("test", "testggss" + jsonArrayNodeFix.optJSONObject(intPosition).optString("nama"));
        strHome = jsonArrayNodeFix.optJSONObject(intPosition).optString("nama");
        return BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN);
    } else {
        Log.i("test", "testggsss" + jsonArrayNodeFix.optJSONObject(intPosition).optString("nama"));
        return BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_RED);
    }
}

```

Gambar 4.27 Fungsi getMarkerFlat(boolean is_pos, int intPosition)

```

public JSONObject execute(String strAwal, String strAkhir){
    strActiveName = strAwal;
    strActiveAkhir = strAkhir;
    endPoint = jsonObjectNodePosition.optInt(strAkhir);

    for (int i = 0; i < arrayListNode.size(); i++) {
        if (arrayListNode.get(endPoint).isActive()){
            traceNode();
            totalWeight = initialWeight;
        } else {
            break;
        }
    }

    traceBack(strAwal, strAkhir);
    JSONArray jsonArrayTemp = new JSONArray();
    for (int i = jsonArrayPath.length() - 1; i >= 0; i--) {
        jsonArrayTemp.put(jsonArrayPath.optString(i));
        if (i == 0){
            jsonArrayPath = jsonArrayTemp;
        }
    }

    JSONObject jsonObjectResult = new JSONObject();
    try {
        jsonObjectResult.put("path", jsonArrayPath);
        jsonObjectResult.put("distance", totalWeight);
    } catch (JSONException e) {
        e.printStackTrace();
    }

    return jsonObjectResult;
}

```

Gambar 4.28 Fungsi execute() dan traceBack()

Nama Fungsi	execute ()
Deskripsi	<p>Fungsi ini digunakan untuk menentukan titik aktif pada peta dan melihat tetangganya. Kemudian mengambil nilai terbaik / terendah yang merupakan hasil perbandingan dari semua titik/ <i>node</i> yang tidak bernilai nol.</p> <p>Lalu fungsi akan pindah ke titik / <i>node</i> terpendek dari hasil perbandingan nilai terbaik sebelumnya dan mengulangi prosesnya. Proses yang diulangi yaitu menentukan titik aktif pada peta dan melihat tetangganya kemudian membandingkan titik – titik / <i>nodes</i> untuk mendapatkan nilai terbaik dari semua titik yang tidak bernilai nol.</p> <p>Proses akan di ulang hingga fungsi menemukan kondisi dimana titik akhir tidak aktif (<i>arrayListNode.get(endPoint).isActive() = false</i>) dan menemukan rutennya.</p> <p>Pada akhir fungsi ini akan dijalankan fungsi traceBack() yang digunakan untuk membalikkan urutan jalur yang telah ditemukan sebelumnya. Fungsi ini terletak didalam fungsi execute(). Lihat gambar 4.28 untuk fungsi execute() dan fungsi traceBack()</p>
Input	Titik awal node (strAwal). Titik strAkhir berasal dari <i>arrayListPos</i> (array yang bervariabel string) yang dipecah menjadi masing masing strAkhir. Sehingga apabila didalam <i>arrayListPos</i> terdapat 3 Pos maka proses dijkstra akan di lakukan sebanyak 3 kali ke masing target pos tersebut. <i>jsonObjectNodePosition</i> , salah satu input untuk fungsi execute(), merupakan json object yang berisikan key value informasi posisi index node pada <i>arrayListNode</i> , <i>arrayListNode</i> , merupakan array yang berisikan varible Node.
Output	JsonObjectResult, merupakan json object yang berisikan key path dan value jsonArrayPath. jsonArrayPath merupakan json array dari varible node yang membentuk suatu rute.

	jsonArrayPath memiliki key distance dan value totalWeight didalamnya. totalWeight merupakan variabel integer dari total jarak node awal ke node akhir.
Status	Succeed

Tabel 4.4 Fungsi execute()

4.2.2 *Blackbox Testing*

Metode ujicoba *black box* memfokuskan pada keperluan fungsional dari aplikasi yang ingin diujikan. Sehingga metode uji aplikasi menggunakan *black box* memungkinkan pengembang aplikasi untuk membuat kondisi input yang akan memenuhi seluruh syarat-syarat fungsional suatu program. *Black box* merupakan pendekatan yang melengkapi untuk menemukan kesalahan aplikasi / *software* selain menggunakan metode *white box* (Mustaqbal, 2015). Ujicoba blackbox berusaha untuk menemukan kesalahan dalam beberapa kategori, diantaranya :

1. Fungsi-fungsi yang salah atau hilang
2. Kesalahan antarmuka (*Interface Errors*)
3. Kesalahan dalam struktur data atau akses database eksternal
4. Kesalahan performa (*Performance Errors*)
5. Kesalahan inisialisasi dan terminasi

Hasil dari uji aplikasi OAM menggunakan metode *black box* akan dibagi menjadi 2 tabel, tabel uji aplikasi android yang digunakan oleh masyarakat (*user*) dan *website* OAM yang digunakan oleh admin ambulans, sebagai berikut :

Tabel 4.22 Tabel Hasil Pengujian Aplikasi Menggunakan Metode *Black Box* Untuk Aplikasi OAM di Android

Masyarakat (<i>User</i>) Yang Menggunakan Aplikasi OAM Pada Android					
No	Fungsi	Input	Hasil yang diharapkan	Hasil Uji	Keterangan
1	Membuka aplikasi	Mengklik aplikasi	Muncul halaman	Muncul halaman pertama yang	Berhasil

			pertama yang menampilkan <i>log in</i> dan <i>register</i>	menampilkan <i>log in</i> dan <i>register</i>	
2	Melakukan pendaftaran / <i>register</i>	Memasukkan informasi berupa nama, nomor hp, <i>email</i> dan <i>password</i> .	Muncul <i>popup box</i> bahwa pendaftaran telah berhasil dilakukan.	Muncul <i>popup box</i> bahwa pendaftaran telah berhasil dilakukan.	Berhasil
3	Melakukan <i>log in</i>	Memasukkan informasi berupa <i>email</i> dan <i>password</i> yang sesuai saat melakukan pendaftaran awal. Klik “ <i>Log in</i> ”	Aplikasi Menampilkan halaman <i>main menu</i> yang terdapat tombol – tombol pesan ambulans, akun saya, dan <i>log out</i> .	Aplikasi Menampilkan halaman <i>main menu</i> yang terdapat tombol – tombol pesan ambulans, akun saya, dan <i>log out</i> .	Berhasil
4	Melakukan Pemesanan Ambulans	Menekan tombol pesan ambulans pada menu utama aplikasi	Aplikasi menampilkan halaman form pemesanan yang terdapat <i>fieldbox</i> untuk alamat, patokan alamat, dan deskripsi pasien.	Aplikasi menampilkan halaman form pemesanan yang terdapat <i>fieldbox</i> untuk alamat, patokan alamat, dan deskripsi pasien.	Berhasil
5	Lanjutkan Pemesanan Ambulans	Masukkan informasi berupa alamat, patokan alamat, dan deskripsi penyakit pasien, lalu	Aplikasi menampilkan halaman pencarian rute terdekat menggunakan Algoritma Dijkstra.	Aplikasi menampilkan halaman pencarian rute terdekat menggunakan Algoritma Dijkstra.	Berhasil

		klik tombol “lanjutkan pesanan”			
6	Lanjutkan Pemesanan Ambulans dengan algoritma Dijkstra	Klik “Find Path” dua kali.	Aplikasi akan menampilkan pos terdekat dari posisi pemesan (<i>user</i>). Aplikasi akan memunculkan <i>pop out</i> yang mengatakan bahwa aplikasi sudah menerima orderannya.	Aplikasi akan menampilkan pos terdekat dari posisi pemesan (<i>user</i>). Aplikasi akan memunculkan <i>pop out</i> yang mengatakan bahwa aplikasi sudah menerima orderannya.	Berhasil
7	Melakukan <i>edit</i> nama pada halaman Akun Saya	Klik <i>textbox</i> nama dan ubah nama yang diinginkan ->Klik “Ubah data”.	Aplikasi akan menampilkan langsung data yang telah diubah.	Aplikasi akan menampilkan langsung data yang telah diubah.	Berhasil
8	Melakukan <i>edit</i> nomor hp pada halaman Akun Saya	Klik <i>textbox</i> nomor hp dan ubah nomor hp yang diinginkan ->Klik “Ubah data”.	Aplikasi akan menampilkan langsung data yang telah diubah.	Aplikasi akan menampilkan langsung data yang telah diubah.	Berhasil
9	Melakukan <i>edit password</i> pada halaman Akun Saya	Klik <i>textbox</i> <i>password</i> dan ubah nama yang diinginkan ->Klik “Ubah data”.	Aplikasi akan menampilkan langsung data yang telah diubah.	Aplikasi akan menampilkan langsung data yang telah diubah.	Berhasil
10	Melakukan <i>edit</i> alamat <i>email</i> pada	Klik <i>textbox</i> alamat <i>email</i> dan ubah	Aplikasi akan menampilkan langsung data	Aplikasi akan menampilkan langsung data	Berhasil

	halaman Akun Saya	alamat <i>email</i> yang diinginkan >Klik “Ubah data”.	yang telah diubah.	yang telah diubah.	
11	Keluar dari akun aplikasi OAM menggunakan fungsi <i>log out</i>	Klik “ <i>log out</i> ”	Aplikasi akan menampilkan halaman pertama aplikasi yaitu halaman <i>log in</i> dan <i>register</i> yang menandakan bahwa pengguna telah keluar dari akun aplikasi.	Aplikasi akan menampilkan halaman pertama aplikasi yaitu halaman <i>log in</i> dan <i>register</i> yang menandakan bahwa pengguna telah keluar dari akun aplikasi.	Berhasil

Tabel 4.23 Tabel Hasil Pengujian Aplikasi Menggunakan Metode *Black Box* Untuk *Website Admin Ambulans OAM*

Admin Ambulans menggunakan OAM berbasis <i>website</i>					
No	Fungsi	Input	Hasil yang diharapkan	Hasil Uji	Keterangan
1	Membuka <i>website</i>	Membuka <i>website</i>	Muncul halaman pertama yang menampilkan <i>log in</i>	Muncul halaman pertama yang menampilkan <i>log in</i>	Berhasil
2	Melakukan <i>log in</i>	Memasukkan informasi berupa <i>email</i> dan <i>password</i> yang sesuai. Klik <i>Log in</i>	<i>Website</i> menampilkan halaman <i>main menu</i> . Dalam <i>main menu</i> terdapat informasi jumlah order, jumlah pengguna	<i>Website</i> menampilkan halaman <i>main menu</i> . Dalam <i>main menu</i> terdapat informasi jumlah order, jumlah pengguna aplikasi, jumlah	Berhasil

			<p>aplikasi, jumlah pos ambulans, dan jumlah total ambulans. Selain itu, di <i>main menu</i> juga dapat dilihat informasi nama seluruh pos – pos agd 118 beserta alamat dan nomor teleponnya..</p>	<p>pos ambulans, dan jumlah total ambulans. Selain itu, di <i>main menu</i> juga dapat dilihat informasi nama seluruh pos – pos agd 118 beserta alamat dan nomor teleponnya..</p>	
3	Melihat order pemesanan ambulans	Klik order > All	<p><i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.</p>	<p><i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.</p>	Berhasil
4	Melihat order pemesanan ambulans	Klik order > <i>Handled</i>	<p><i>Website</i> akan menampilkan informasi pemesanan dengan status <i>handled</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.</p>	<p><i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.</p>	Berhasil
5	Melihat order pemesanan ambulans	Klik order > <i>Unhandled</i>	<p><i>Website</i> akan menampilkan informasi pemesanan</p>	<p><i>Website</i> akan menampilkan seluruh informasi pemesanan</p>	Berhasil

			dengan status <i>unhandled</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	
6	Melihat order pemesanan ambulans	Klik order > <i>Out Of Coverage</i>	<i>Website</i> akan menampilkan informasi pemesanan dengan status <i>out of coverage</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	<i>Website</i> akan menampilkan informasi pemesanan dengan status <i>out of coverage</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	Berhasil
7	Melihat order pemesanan ambulans	Klik order > <i>Declined</i>	<i>Website</i> akan menampilkan informasi pemesanan dengan status <i>declined</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	<i>Website</i> akan menampilkan informasi pemesanan dengan status <i>declined</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	Berhasil
8	Melihat pos - pos ambulans AGD 118	Klik AGD Stations	<i>Website</i> akan menampilkan informasi pos - pos ambulans di Jakarta berupa nama pos, alamat, titik	<i>Website</i> akan menampilkan informasi pos - pos ambulans di Jakarta berupa nama pos, alamat, titik	Berhasil

			longitude dan latitude, serta nomor teleponnya.	longitude dan latitude, serta nomor teleponnya.	
9	Melihat informasi pengguna terdaftar aplikasi OAM	Klik Users	<i>Website</i> akan menampilkan informasi pengguna terdaftar aplikasi berupa nama, nomor hp, dan alamat email.	<i>Website</i> akan menampilkan informasi pengguna terdaftar aplikasi berupa nama, nomor hp, dan alamat email.	Berhasil
10	Keluar dari akun website OAM dengan fitur <i>log out</i>	Klik admin > <i>log out</i>	<i>Website</i> akan menampilkan halaman <i>log in</i> sebagai tanda admin sudah keluar dari akunnya.	<i>Website</i> akan menampilkan halaman <i>log in</i> sebagai tanda admin sudah keluar dari akunnya.	Berhasil

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini dilakukan berawal dari tahap studi literatur hingga sampai ke tahap akhir ini, yaitu penulisan laporan. Setelah melalui serangkaian proses pelaksanaan penelitian, dapat disimpulkan hasil dari penelitian ini terkait tujuan dari penelitian, yaitu :

1. Penulis berhasil menghasilkan perancangan aplikasi OAM secara konseptual dan realisasinya yang bertujuan untuk membantu masyarakat (*user*) Kota DKI Jakarta sebagai pengguna aplikasi untuk dapat melakukan pemesanan ambulans. Penulis telah membuat rancangan aplikasi OAM yang dibahas lengkap pada BAB III pada fase desain aplikasi dari hasil identifikasi sebelumnya. Desain alur dan cara kerja aplikasi telah direpresentasikan dengan UML diagram yaitu *Use Case Diagram*, *Activity Diagram*, *Class Diagram*, *Entity Relationship Diagram* (ERD), *Logical Database Model*, *Physical Database Model*, serta desain *User Interfacenya*.
2. Penulis berhasil membangun aplikasi OAM yang berjalan dengan baik sehingga dapat digunakan untuk membantu masyarakat (*user*) Kota DKI Jakarta sebagai pengguna aplikasi untuk dapat melakukan pemesanan ambulans. Penulis juga membuat *website* admin ambulans yang dapat membantu dalam proses pemesanan ambulans dan menentukan posisi pos ambulans terdekat dari pemesan menggunakan Algoritma Dijkstra yang telah dijalankan pada aplikasi Android OAM.

5.2 Saran

Penelitian yang dilkakukan tentunya tidak akan lepas dari kekurangan dan kelemahan. Oleh karena itu, untuk membuat sistem aplikasi OAM yang lebih baik maka ada beberapa hal yang perlu diperhatikan, diantaranya adalah :

1. Pada penelitian aplikasi pemesanan ambulans OAM ini, penulis masih memiliki beberapa keterbatasan terutama untuk menginput titik-titik rute (*nodes*) yang dapat mengcover seluruh lokasi yang ada di DKI Jakarta. Memperbanyak titik – titik rute (*nodes*) yang ada didalam peta sehingga rute lebih terincikan dan memperbanyak posibilitas rute yang ada tentu dapat membantu aplikasi menjadi lebih baik dalam eksekusinya.
2. Di dalam kasus penelitian ini, rute terpendek yang dihasilkan oleh sistem melalui Algoritma Dijkstra belum tentu menghasilkan rute yang efisien jika dibandingkan dengan kasus nyata. Hal tersebut dipengaruhi oleh berbagai faktor seperti kemacetan, cuaca, hari tertentu dan faktor lainnya. Sehingga diharapkan adanya pengembangan sistem yang tidak hanya melakukan pendekatan dari segi rute terpendek, namun juga menggunakan metode dan pendekatan lainnya agar tingkat efisiensi rute lebih optimal.