

**RANCANG BANGUN APLIKASI OAM (*ORDER AMBULANCE MOBILE*) MENGGUNAKAN GIS (*GEOGRAPHIC INFORMATION SYSTEM*) DAN ALGORITMA DIJKSTRA**

**TUGAS AKHIR**



**ATIKAH CHAIRUNNISA**

**1112001018**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNIK DAN ILMU KOMPUTER  
UNIVERSITAS BAKRIE  
JAKARTA  
2017**

**RANCANG BANGUN APLIKASI OAM (*ORDER AMBULANCE MOBILE*) MENGGUNAKAN GIS (*GEOGRAPHIC INFORMATION SYSTEM*) DAN ALGORITMA DIJKSTRA**

**TUGAS AKHIR**

**Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer**



**ATIKAH CHAIRUNNISA**

**1112001018**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS TEKNIK DAN ILMU KOMPUTER  
UNIVERSITAS BAKRIE  
JAKARTA  
2017**

## **HALAMAN PERNYATAAN ORISINALITAS**

**Tugas Akhir ini adalah hasil karya saya sendiri,  
dan semua sumber baik yang dikutip maupun dirujuk  
telah saya nyatakan dengan benar.**

**Nama : Atikah Chairunnisa**

**NIM : 1112001018**

**Tanda Tangan :**

**Tanggal : 30 Januari 2017**

## **HALAMAN PENGESAHAN**

Tugas Akhir ini diajukan oleh:

Nama : Atikah Chairunnisa  
NIM : 1112001018  
Program Studi : Teknik Informatika  
Fakultas : Teknik dan Ilmu Komputer  
Judul Skripsi : Rancang Bangun Aplikasi OAM (*Order Ambulance Mobile*)  
Menggunakan Metode GIS (*Geographic Information System*)  
Dan Algoritma Dijkstra.

**Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai persyaratan yang diperlukan untuk memperoleh gelar Sarjana Komputer (S.Kom) pada Program Studi Informatika Fakultas Teknik dan Ilmu Komputer, Universitas Bakrie.**

## **DEWAN PENGUJI**

Pembimbing : Yusuf Lestanto, S.T., M.Sc. ( )

Penguji 1 : Dr. Siti Rohajawati, S.Kom., M.Kom. ( )

Penguji 2 : Berkah I. Santoso, S.T, M. T.I. ( )

Ditetapkan di : Jakarta

Tanggal : 30 Januari 2017

## **UNGKAPAN TERIMA KASIH**

Alhamdulillahirabbil' alamin, puji syukur kehadirat Allah Subhanahu wa ta'ala yang senantiasa melimpahkan kasih sayang, rahmat, nikmat, dan karunia-Nya sehingga tugas akhir penulis yang berjudul Rancang Bangung Aplikasi OAM (*Order Ambulance Mobile*) Menggunakan GIS (*Geographic Information System*) Dan Algoritma Dijkstra dapat terselesaikan dengan baik. Shalawat serta salam semoga senantiasa tercurahkan kepada Rasulullah Shalallahu Alaihi Wassalam, keluarga dan sahabatnya yang telah membawa umat manusia ke zaman yang penuh dengan cahaya ilmu. Penulisan Tugas Akhir ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Komputer Program Studi Informatika Fakultas Teknik dan Ilmu Komputer Universitas Bakrie. Dalam proses penyelesaian Tugas Akhir ini tidak terlepas dari berbagai hambatan dan kesulitan, namun terdapat banyak pihak yang bersedia meluangkan waktunya untuk membantu serta memberikan dukungan moril untuk penulis. Oleh karena itu, pada kesempatan ini penulis ingin mengungkapkan terima kasih dan penghargaan kepada :

1. Bapak Prof. Dr. Hoga Saragih, S.T., M.T., selaku Kepala Program Studi Informatika, atas segala saran serta dukungan yang diberikan dalam pelaksanaan penelitian ini.
2. Bapak Yusuf Lestanto, S.T., M.Sc., selaku dosen pembimbing, atas segala waktu yang diluangkan. Beliau telah memberikan kesabaran dan pengertian dalam memberikan bimbingan dan arahan untuk penulis. Penulis ucapkan banyak terima kasih kepada beliau karena telah memberikan kesempatan dan membimbing penulis dengan sangat baik.
3. Ibu Dr. Siti Rohajawati, S.Kom., M.Kom. dan Berkah I. Santoso, S.T, M. T.I. selaku dosen penguji, atas segala saran dan koreksi yang membangun untuk membantu meningkatkan kualitas laporan tugas akhir ini. Terima kasih atas ilmu yang diberikan.

4. Suami tercinta, Farly Nur Dewantara, S.T, atas perhatian, kasih sayang, kerja keras dan keringat yang tiada henti diberikan kepada penulis. Serta senantiasa memberikan kepercayaan dan dukungannya hingga tugas akhir ini terselesaikan.
5. Kedua orang tua tercinta, Dr. Amru Sofian, SpOG, KOnk, MWals dan Drg. Suci Nuralitha, atas perhatian dan kasih sayang, dukungan, dan doa yang telah mengantarkan penulis hingga titik ini. Pencapaian ini dipersembahkan untuk kalian.
6. Kedua adik tercinta , Ashil Muhammad Abdul Rasyid dan Athira Nadhila Rizka, yang telah menemani, menjadi teman pengisi hari dengan penuh canda, tawa dan doa.
7. Untuk sahabar dalam kebaikan Addina Nuriyanti Rahmi, Ayyu Andhysa, Ana Ainul Syamsi, Khairunnisah, atas waktu, kebersamaan, canda tawa, bantuan, dukungan, dan doa yang diberikan selama ini.
8. Teman-teman TIF seperjuangan yang telah membantu penulis dengan memberikan dukungan moril, doa, dan informasi untuk penyelesaian tugas akhir ini. Terima kasih atas kebaikan kalian selama lebih dari 4 tahun ini.
9. Seluruh pihak yang telah memberikan dukungan dalam penyelesaian tugas akhir ini yang tidak dapat penulis sebutkan satu-persatu.

Semoga Allah Subhanahu wa ta'ala melimpahkan berkah dan nikmat-Nya serta membalas kebaikan semua pihak yang telah membantu penulis dalam penyusunan tugas akhir ini. Penulis berharap semoga tugas akhir ini dapat memberikan manfaat bagi semua pihak dan dalam dunia pendidikan.

Jakarta, 30 Januari 2017

Atikah Chairunnisa

## **HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI**

Sebagai sivitas akademik Universitas Bakrie, saya yang bertanda tangan di bawah ini:

Nama : Atikah Chairunnisa  
NIM : 1112001018  
Program Studi : Teknik Informatika  
Fakultas : Teknik Dan Ilmu Komputer  
Jenis Tugas Akhir : Rancang Bangun

Demi Pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Bakrie **Hak Bebas Royalti Noneksklusif (Non-exclusive Royalty-Free Right)** atas karya ilmiah saya yang berjudul:

### **Rancang Bangun Aplikasi OAM (*Order Ambulance Mobile*) Menggunakan GIS (*Geographic Information System*) dan Algoritma Dijkstra**

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Bakrie berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta untuk kepentingan akademis.

Demikian pernyataan ini saya buat sebenarnya.

Dibuat di : Jakarta  
Pada tanggal : 30 Januari 2017

Yang Menyatakan

(Atikah Chairunnisa)

**RANCANG BANGUN APLIKASI OAM (*ORDER AMBULANCE MOBILE*)  
MENGGUNAKAN GIS (*GEOGRAPHIC INFORMATION SYSTEM*) DAN  
ALGORITMA DIJKSTRA**

Atikah Chairunnisa

---

**Abstrak**

Lokasi dan waktu pertolongan yang dibutuhkan warga saat mengalami kondisi yang mengancam nyawa seringkali tidak terduga. Ketika hal yang tidak diinginkan terjadi, masyarakat harus melakukan pemesanan ambulans secara manual melalui telepon. Beberapa kendala yang ditemui dalam proses pemesanannya di antaranya adalah *false call*, seseorang yang menghubungi suatu instansi dalam keadaan darurat tidak sesuai dengan kebenaran atau fakta, yang dihadapi oleh instansi penyedia jasa ambulans. Terlebih lagi kemungkinan *human error* sewaktu mengolah data yang dapat terjadi. Kebutuhan masyarakat dan rumah sakit agar dapat melakukan proses ini secara efektif dan cepat di era teknologi seperti sekarang seharusnya dapat diwujudkan. Penelitian ini mengatasi masalah efisiensi pengambilan dan pengolahan data dari pasien menggunakan aplikasi *OAM (Order Ambulance Mobile)*. Aplikasi menggunakan *Geographic Information System (GIS)* yang digunakan masyarakat berbasis *android*. Aplikasi juga akan diimplementasikan dengan algoritma *Dijkstra* dalam memudahkan penentuan pos terdekat dari pasien (*user*) yang melakukan pemesanan.

**Kata Kunci:** *Geographic Information System, Android, OAM, Pemesanan Ambulans, Algoritma Dijkstra*

**DESIGN AND DEVELOPMENT OF OAM (ORDER AMBULANCE MOBILE)  
APPLICATION USING GIS (GEOGRAPHIC INFORMATION SYSTEM) AND  
DIJKSTRA ALGORITHM**

Atikah Chairunnisa

---

**Abstract**

The location and time when the citizens of emergency, experience life-threatening conditions, are often unpredictable. When undesirable things occurs, society are required to order ambulance manually by phone. Some of the obstacles encountered in the process of ordering ambulances are false call, a person who contacted an agency in an emergency but does not correspond to the truth or fact. And chances of human error when inputting the information data is also considered problem for the health instance. The needs of the community and the hospital or health instances in order to carry out this process effectively and quickly in the era of technology as it is now to be realized. This study overcome the problem of retrieval efficiency and data processing applications from patients using OAM (Order Ambulance Mobile). The applications implements Geographic Information System that will be used in android based devices. OAM application will also be implemented with Dijkstra's algorithm to facilitate the determination of the nearest posts from the patient (user) who placed the order.

**Key Words:** *Geographic Information System, Android, OAM, Order Ambulance, Dijkstra Algorithm*

## DAFTAR ISI

<b>HALAMAN PERNYATAAN ORISINALITAS.....</b>	<b>iii</b>
<b>HALAMAN PENGESAHAN.....</b>	<b>iv</b>
<b>UNGKAPAN TERIMA KASIH.....</b>	<b>v</b>
<b>HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI.....</b>	<b>vii</b>
<b>ABSTRAK.....</b>	<b>viii</b>
<b>ABSTRACT.....</b>	<b>ix</b>
<b>DAFTAR ISI.....</b>	<b>x</b>
<b>DAFTAR GAMBAR.....</b>	<b>xii</b>
<b>DAFTAR TABEL.....</b>	<b>xv</b>
<b>DAFTAR LAMPIRAN.....</b>	<b>xvi</b>
<b>DAFTAR SINGKATAN.....</b>	<b>xvii</b>
<b>BAB I. PENDAHULUAN.....</b>	<b>1</b>
1.1    Latar Belakang.....	1
1.2    Rumusan Masalah.....	4
1.3    Batasan Masalah.....	5
1.4    Tujuan Penelitian.....	5
1.5    Manfaat Penelitian.....	6
1.6    Sistematika Penulisan.....	6
<b>BAB II. LANDASAN TEORI.....</b>	<b>8</b>
2.1    Penelitian Terdahulu.....	8
2.2    GIS ( <i>Geographic Information System</i> ).....	10
2.2.1    Komponen GIS.....	11
2.2.2 <i>Geocoding</i> dan <i>Geocoder</i> .....	17
2.3    Android.....	18
2.3.1    Anatomi Android.....	20
2.4    Algoritma Pencarian Jarak Terpendek <i>(Shortest Path Algorithm}</i> .....	23

2.4.1	Algoritma Dijkstra.....	25
2.4.2	Algoritma Bellman Ford.....	27
2.4.3	Algoritma Floyd Warshall.....	28
2.4.4	Perbandingan Algoritma Dijkstra, Bellman Ford, dan Floyd Warshall.....	29
2.5	<i>System Development Life Cycle (SDLC)</i> .....	31
<b>BAB III. METODE PENELITIAN.....</b>		<b>37</b>
3.1	Perancangan Aplikasi.....	37
3.2	Jenis Penelitian.....	53
3.3	Objek Penelitian.....	53
<b>BAB IV. IMPLEMENTASI DAN PENGUJIAN APLIKASI.....</b>		<b>54</b>
4.1	Implementasi.....	54
4.1.1	Implementasi Aplikasi OAM pada Android.....	54
4.1.2	Implementasi Aplikasi OAM pada <i>Website Admin</i> .....	62
4.2	Pengujian Aplikasi OAM.....	67
4.2.1	<i>White Box Testing</i> .....	75
4.2.2	<i>Black Box Testing</i> .....	75
<b>BAB V. KESIMPULAN DAN SARAN.....</b>		<b>76</b>
5.1	Kesimpulan.....	76
5.2	Saran.....	76
<b>DAFTAR PUSTAKA.....</b>		<b>78</b>

## DAFTAR TABEL

Tabel 2.1	Penelitian –Penelitian Terdahulu.....	9
Tabel 2.2	Bentuk-bentuk data informasi lokasi (spasial) .....	13
Tabel 2.3	Perbandingan <i>online geocoding API</i> .....	17
Tabel 2.4	Algoritma Djikstra dalam bentuk <i>Pseudocode</i> .....	26
Tabel 2.5	Algoritma Bellman Ford.....	28
Tabel 2.6	Algoritma Floyd Warshall.....	29
Tabel 2.7	Perbandingan algoritma pencari jarak terpendek <i>(Shortest Path Algorithm)</i> .....	30
Tabel 2.8	Perbandingan model-model SDLC ( <i>Waterfall</i> , <i>2-1 Process</i> , dan <i>V-Process</i> ).....	32
Tabel 3.1	Deskripsi <i>Use Case Register</i> .....	37
Tabel 3.2	Deskripsi <i>Use Case Log In</i> .....	37
Tabel 3.3	Deskripsi <i>Use Case Log In (Website Admin)</i> .....	38
Tabel 3.4	Deskripsi <i>Use Case Log Out</i> .....	39
Tabel 3.5	Deskripsi <i>Use Case Log Out (Website Admin)</i> .....	39
Tabel 3.6	Deskripsi <i>Use Case Melihat Informasi Akun</i> .....	40
Tabel 3.7	Deskripsi <i>Use Case Memperbarui Akun</i> .....	40
Tabel 3.8	Deskripsi <i>Use Case Pesan Ambulans</i> .....	41
Tabel 3.9	Deskripsi <i>Use Case Melanjutkan Pemesanan</i> .....	41
Tabel 3.10	Deskripsi <i>Use Case Menentukan Pos Terdekat Menggunakan</i> Algoritma Dijkstra.....	42

## DAFTAR GAMBAR

Gambar 2.1	Kategori <i>Geographic Information System</i> .....	11
Gambar 2.2	Komponen – Komponen GIS.....	12
Gambar 2.3	Vektor dan Raster data model.....	15
Gambar 2.4	Anatomi Android.....	20
Gambar 2.5	Klasifikasi graf berdasarkan orientasinya; (a) Graf terarah / <i>Directed graph</i> (b) Graf tidak terarah / <i>Undirected graph</i> .....	24
Gambar 2.6	Diagram proses pada <i>System Life Cycle Development</i> (SDLC) model <i>Waterfall</i> (Pressman,1997) .....	33
Gambar 3.1	<i>Use Case Diagram</i> Aplikasi OAM.....	40
Gambar 3.2	<i>Activity Diagram</i> pada Aplikasi OAM.....	41
Gambar 3.3	<i>Class Diagram</i> pada Aplikasi OAM.....	43
Gambar 3.4	<i>Entity Relationship Diagram</i> (ERD) pada Aplikasi OAM.....	45
Gambar 3.5	<i>Logical Database Model</i> pada Aplikasi OAM.....	46
Gambar 3.6	<i>Physical Database Model</i> pada Aplikasi OAM.....	47
Gambar 3.7	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Satu.....	48
Gambar 3.8	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Dua.....	48
Gambar 3.9	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Tiga.....	49
Gambar 3.10	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Empat.....	49
Gambar 3.11	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Lima.....	50
Gambar 3.12	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Enam.....	50

Gambar 4.1	Tampilan Antarmuka Android Halaman <i>Register</i> .....	55
Gambar 4.2	Tampilan Antarmuka Android Halaman <i>Log In</i> .....	55
Gambar 4.3	Tampilan Antarmuka Android Menu Utama.....	56
Gambar 4.4	Tampilan Antarmuka Android Pesan Ambulans.....	57
Gambar 4.5	Tampilan Antarmuka Android Pesan Ambulans( <i>filled</i> ).....	58
Gambar 4.6	Tampilan Antarmuka Android Pesan Ambulans dengan Algoritma Dijkstra.....	59
Gambar 4.7	Tampilan Antarmuka Android Pesan Ambulans dengan Algoritma Dijkstra Dua.....	59
Gambar 4.8	Tampilan Antarmuka Android Pesan Ambulans dengan Algoritma Dijkstra Tiga.....	60
Gambar 4.9	Tampilan Antarmuka Android Pesan Ambulans dengan Algoritma Dijkstra telah terkonfirmasi.....	61
Gambar 4.10	Tampilan Antarmuka <i>Website OAM</i> Halaman <i>Log In Admin</i> .....	63
Gambar 4.11	Tampilan Antarmuka <i>Website OAM</i> Halaman <i>Main Menu</i> .....	63
Gambar 4.12	Tampilan Antarmuka <i>Website OAM</i> Halaman <i>Main Menu Dua</i> .....	64
Gambar 4.13	Tampilan Antarmuka <i>Website OAM</i> Menu <i>Sidebar</i> .....	65
Gambar 4.14	Tampilan Antarmuka <i>Website OAM</i> Halaman Pemesanan Ambulans Seluruh Status.....	66
Gambar 4.15	Tampilan Antarmuka <i>Website OAM</i> Halaman <i>AGD Stations</i> .....	66
Gambar 4.16	Tampilan Antarmuka <i>Website OAM</i> Halaman <i>Users</i> .....	67
Gambar 4.17	Tampilan Antarmuka Android Halaman Pesan Ambulans( <i>filled</i> ).....	68
Gambar 4.18	Tampilan Antarmuka Android Halaman Pesan Ambulans.....	69
Gambar 4.19	Sebagian Kodingan pada Android Studio.....	70
Gambar 4.20	Sebagian Kodingan pada Android Studio Dua.....	71
Gambar 4.21	Sebagian Kodingan pada Android Studio Tiga.....	71
Gambar 4.22	Sebagian Kodingan pada Android Studio Empat.....	72

Gambar 4.23 Sebagian Kodingan pada Android Studio Lima.....	73
Gambar 4.24 Tampilan Antarmuka Android Pesan Ambulans.....	74

## **DAFTAR LAMPIRAN**

Lampiran 1. Wawancara .....	84
Lampiran 2. <i>Software Requirement System</i> (SRS) .....	87
Lampiran 3. Elisitasi Kebutuhan ( <i>Requirement Elicitation</i> ).....	95
Lampiran 4. Deskripsi <i>Use Case Diagram</i> Aplikasi OAM .....	99
Lampiran 5. Tabel Hasil Pengujian Aplikasi OAM Menggunakan Metode <i>White Box Testing</i> .....	112
Lampiran 6. Tabel Hasil Pengujian Aplikasi OAM Menggunakan Metode <i>Black Box Testing</i> .....	117

## **DAFTAR SINGKATAN**

ADT	<i>Android Development Tools</i>
AGD	Ambulans Gawat Darurat
API	<i>Application Program Interface</i>
DBMS	<i>Database Management System</i>
ERD	<i>Entity Relationship Diagram</i>
GIS	<i>Geographic Information System</i>
GPS	<i>Global Positioning System</i>
JDK	<i>Java Development Kit</i>
KMP	<i>Knutt Morris Pratt</i>
MADLC	<i>Mobile Application Development Life Cycle</i>
SDLC	<i>System Development Life Cycle</i>
OS	<i>Operating System</i>
PC	<i>Personal Computer</i>
SDK	<i>Software Development Kit</i>
UI	<i>User Interface</i>
UML	<i>Unified Model Language</i>

## 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.

Waktu yang dibutuhkan dalam penanganan pasien (*response time*) dari saat operator menerima telefon hingga sampai ketempat tujuan, 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* 15 hingga 20 menit. Hal ini menunjukkan bahwa *response time* merupakan catatan yang penting dan sulit

ditekan (Nugroho, 2016). Selain itu admin AGD 118, mengalami beberapa kendala dalam memetakan posisi pemesan ambulans yang sebenarnya. Proses pemetaan posisi pemesan masih dilakukan secara manual sehingga penetuan pos ambulans terdekat juga berdasarkan perkiraan dan bukan perhitungan yang sebenarnya. Kepanikan dan suara telepon oleh pemesan juga terkadang mengganggu admin dalam mendeskripsikan ulang laporan pemesanannya. 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 pos - pos ambulans serta 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 tertentu?

### 1.3 Batasan Masalah

Batasan dari penelitian ini adalah sebagai berikut:

1. Aplikasi berjalan pada O.S Android dengan versi minimum 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 yang terpasang pada *smartphone*.
5. Penelitian tidak membahas tentang *tracking* pasien (*user*) maupun *driver* ambulans.
6. Penelitian tidak mencakup jaringan dan keamanan (*security*) pada *device*.
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 tertentu.

## 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 tertentu.

## 1.6 Sistematika Penulisan

Sistematika penulisan tugas akhir ini adalah sebagai berikut:

### BAB I PENDAHULUAN

Bab ini berisi latar belakang, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian serta sistematika penulisan.

### BAB II LANDASAN TEORI

Bab ini berisi analisis penelitian terdahulu dan beberapa penjelasan mengenai GIS (*Geographic Information System*), android, algoritma pencarian jarak terpendek (*shortest path algorithm*), dan SDLC (*System Development Life Cycle*).

### BAB III METODE PENELITIAN

Bab ini berisi perancangan aplikasi menggunakan SDLC(*System Development Life Cycle*), jenis penelitian, dan objek penelitian.

### BAB IV IMPLEMENTASI DAN PENGUJIAN APLIKASI

Bab ini berisi implementasi aplikasi OAM pada android dan *website*

*Admin*, serta pengujian aplikasi OAM menggunakan metode *white box testing* dan metode *black box testing*.

**BAB V KESIMPULAN DAN SARAN**

Bab ini berisi beberapa kesimpulan dan saran dari hasil penelitian.

## 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 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 dikhususka 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 dibangun menggunakan bahasa pemrograman PHP dan *database* mySQL. (Fauzi, 2011).
2. 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 terdapatnya banyaknya penelpon yang tidak bertanggung jawab terjadi pada Damkar Bandung, maka dibutuhkan aplikasi yang dapat membantu melakukan validasi 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)

3. Penelitian yang dilakukan oleh Stevian Suryo Saputro dengan judul *Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata di Kota Manado berbasis Mobile 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 dibangun menggunakan metode GIS dan algoritma Dijkstra (Saputro, 2013).

**Tabel 2.1 Penelitian – Penelitian Terdahulu**

Judul	Pengarang	Keterangan
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 <i>On-Call Positioning</i> Menggunakan GIS ( <i>Geographic Information System</i> ) dan GPS pada Dinas Pemadam Kebakaran Bandung	Farly Nur Dewantara (Dewantara, 2013)	Aplikasi pelaporan pemadam kebakaran dengan memanfaatkan lokasi pemesan dan foto sebagai bukti laporan. Aplikasi dibatasi hanya untuk wilayah bandung dan berbasis android menggunakan GIS.

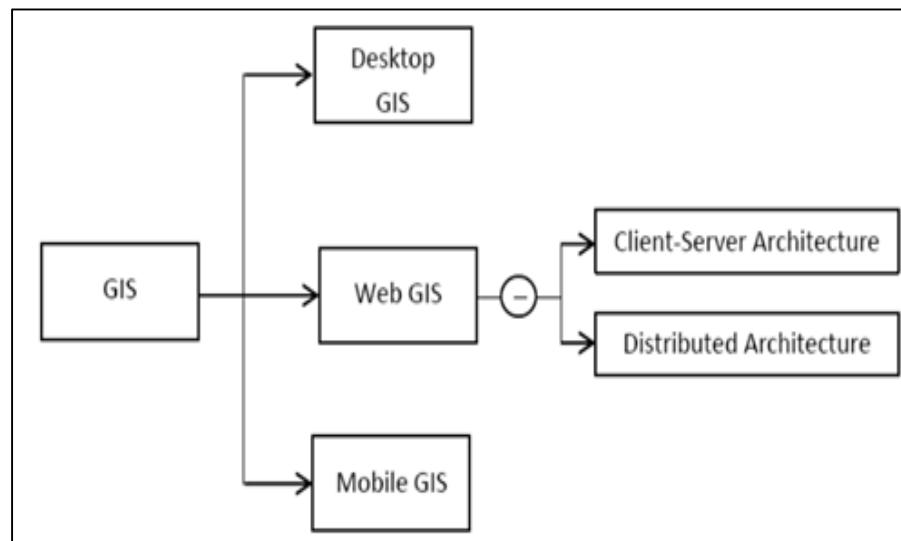
Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata di Kota Manado berbasis <i>Mobile Web</i> Dengan Algoritma Dijkstra.	Stevian Suryo Saputro (Saputro, 2013)	Aplikasi berbasis mobile web yang dapat menampilkan rute antar satu tempat wisata dengan tempat wisata lainnya, serta rute dari posisi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

## 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 beraserensi geografis (Riyanto, 2010). Sistem ini berkaitan erat dengan informasi geografis sehingga terdapat pendapat lain yang juga 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

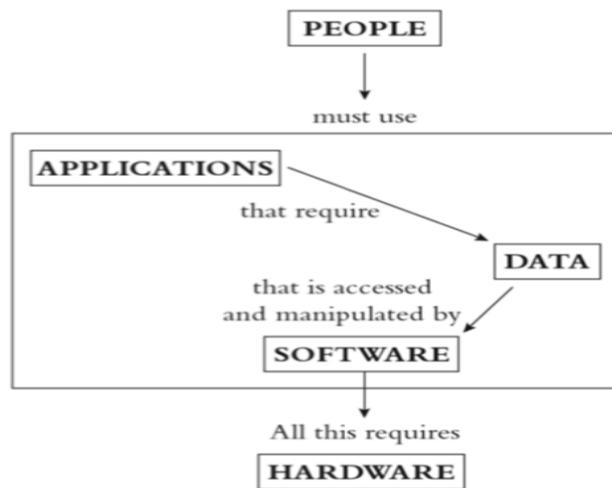
untuk menampilkan data peta, analisis data, dan mempublikasi (*publish*) data. GIS berbasis *web* (*Web GIS*) merupakan sistem informasi geografis yang didistribusikan pada seluruh lingkungan jaringan komputer untuk melakukan integrasi, penyebaran, dan melakukan komunikasi informasi geografis secara *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)**

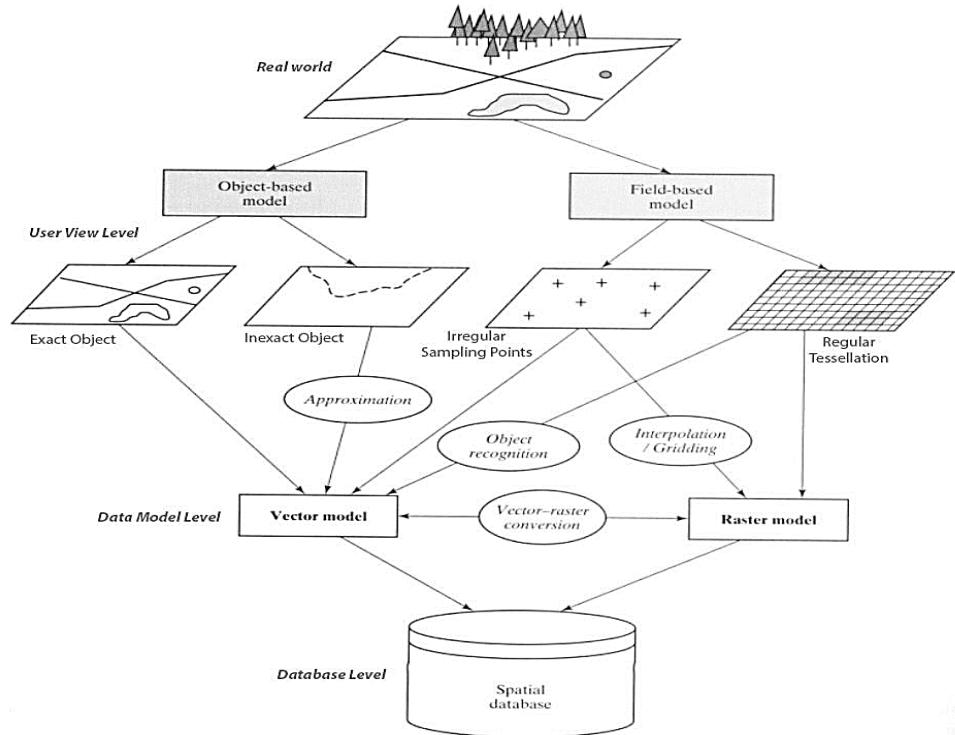
Bentuk-bentuk informasi lokasi	Deskripsi	Format Titik
<b>Titik (Dot)</b>	Bentuk data ini merupakan bentuk data paling sederhana bagi obyek spasial. Tipe data ini tidak memiliki dimensi.  Contoh : Lokasi kecelakaan	Format titik : Koordinat tunggal. Tanpa panjang/jarak. Tanpa luas
<b>Garis (Polyline)</b>	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.

<b>Area (<i>Polygon</i>)</b>	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.
<b>Permukaan (3D)</b>	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 terdapat 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.



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

Dalam data spasial, data dapat direpresentasikan ke dalam dua model yaitu:

### 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 dari 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)**

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

<b>(Data resources)</b>		Map Data Sciences	
<b>Return Format</b>	JSON/XML	JSON/XML	XML
<b>Kapasitas request (Request capacity)</b>	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
<b>Presisi geocoding (Geocoding precision)</b>	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

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 terdapat 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*)

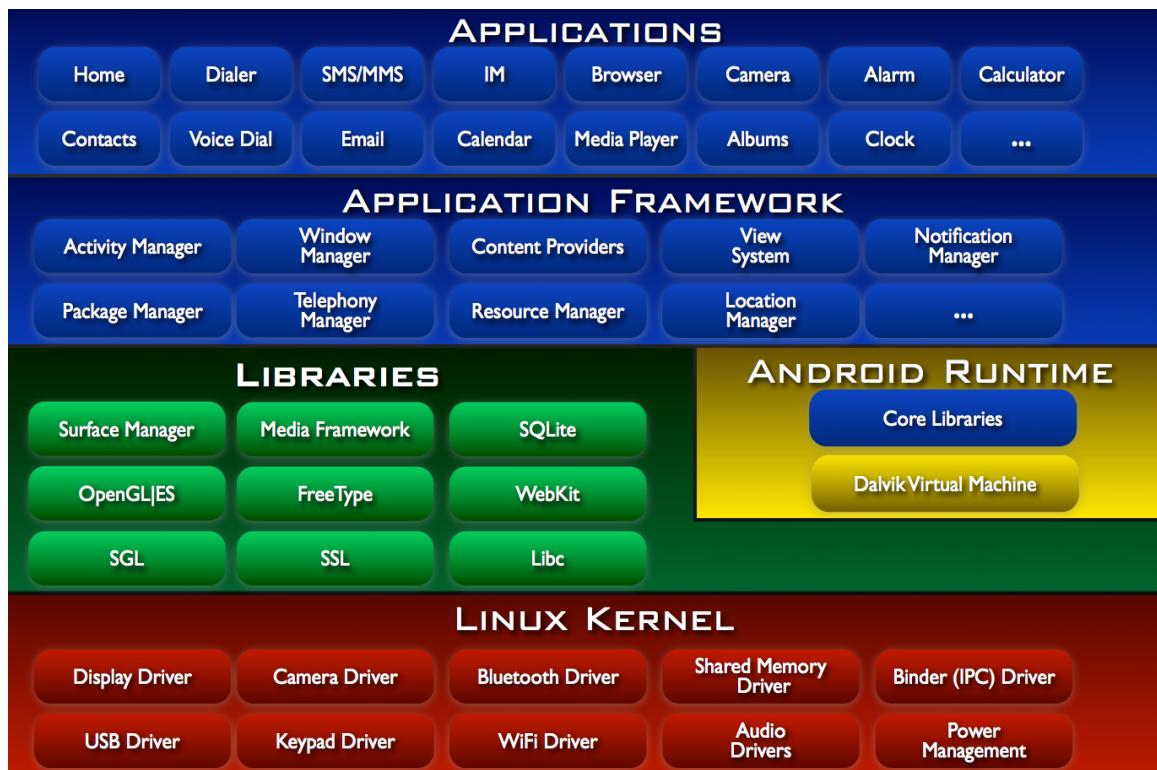
*Platform* android disediakan melalui lisensi *open source*. Hal itu berarti, pengembang dapat dengan bebas untuk mengembangkan aplikasi. Aplikasi dengan lisensi *open source* biasanya lebih stabil, tidak memakan biaya yang banyak / tanpa biaya, memiliki komunitas (*support community*) yang besar sehingga 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 terdapat 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

Android merupakan sistem operasi yang didesain berbasis Linux dan dikostumisasi untuk dapat berjalan pada perangkat *mobile*. Meskipun android diciptakan menggunakan Linux kernel, namun android bukanlah linux. Linux kernel menyediakan service sistem inti untuk android android seperti memori manajemen, memori process dan *tasks*, manajemen *power*, *networking stack* dan berbagai jenis *device driver* yang akan melakukan interaksi dengan *hardware* (Panigrahy, 2015). Linux juga mendukung *shared libraries* dan

merupakan lisensi *opensource*. Kernel linux menyediakan driver layar, kamera, keypad, *Flash Memory*, IPC (*Interprocess Communication*) dan audio untuk mengatur aplikasi dan keamanannya.

## 2. Native Libraries

Android menggunakan beberapa paket pustaka yang terdapat pada C/C++ yang memiliki berbagai jenis tipe *service*. *Library* ini didominasi dengan *open source community* (Panigrahy, 2015). Dengan standar *Berkeley Software Distribution* (BSD) hanya setengah dari yang aslinya untuk tertanam pada kernel Linux. Beberapa pustaka diantaranya adalah :

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

## 3. *Android Runtime*

Aplikasi Android berjalan pada *Dalvic Virtual Machine* (Dalvic VM). Dalvic VM memiliki kemiripan dengan Java VM namun telah dioptimisasi untuk beberapa *devices* yang memiliki keterbatasan kapasitas memproses memori (Panigrahy, 2015). Dalvic VM merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvic hanyalah interpreter mesin virtual yang mengeksekusi *file* dalam format Dalvic Executable (\*.dex). Dalvik *virtual machine* dapat mendukung banyak pemrosesan pada satu *device* karena menggunakan *runtime memory* secara sangat efisien. Meskipun Dalvic VM

berjalan dengan baik, terkadang terdapat beberapa *lag* yang cukup besar ketika aplikasi dijalankan. Dan saat itu lah *virtual machine* baru bernama ART datang. ART VM merupakan *virtual machine* baru yang dikenalkan pada Android 4.4 (KitKat) dan diimplementasikan sepenuhnya pada Android 5.0 (Lollipop). ART VM memiliki konsep *Ahead Of Time* (AOT). Ini berarti aplikasi baru di *compiled* pada saat instalasi, bahkan sebelum mereka *launched*.

#### 4. *Application Framework*

*Application framework* merupakan bagian dari platform android yang banyak tidak diketahui oleh *developer*. *Application framewotk* memberikan keleluasaan untuk membuat tampilan *user interface*, interaksi dengan kapabilitas *device* seperti kamera, *location services*, dan performa banyak hal berguna lainnya (Panigrahy, 2015). Beberapa bagian penting dalam kerangka aplikasi Android adalah sebagai berikut :

##### a. *Activity Manager*

Berfungsi untuk mengontrol siklus hidup aplikasi (*activity life cycle*), manajemen dan menjaga keadaan *BackStack* untuk navigasi penggunaan.

##### b. *View System*

Berfungsi untuk memberikan kontrol akses terhadap tampilan *user interface* dalam membangun tampilan *user interface* aplikasi.

##### c. *Content Providers*

Berfungsi untuk memberikan tampilan untuk melakukan publikasi dan *share* data diantara aplikasi dengan aplikasi lainnya.

##### d. *Resource Manager*

Berfungsi untuk mengatur akses terhadap sumber daya yang terdapat dalam program seperti *user interface layout*, *strings*, *colors*, *dimensions*, dan banyak hal lainnya.

d. *Location Manager*

Berfungsi untuk menyediakan informasi detail mengenai sistem lokasi.

e. *Notification Manager*

Berfungsi untuk memanajemen peringatan (*reminder*) dan notifikasi seperti, pesan masuk, janji, dan lain lain.

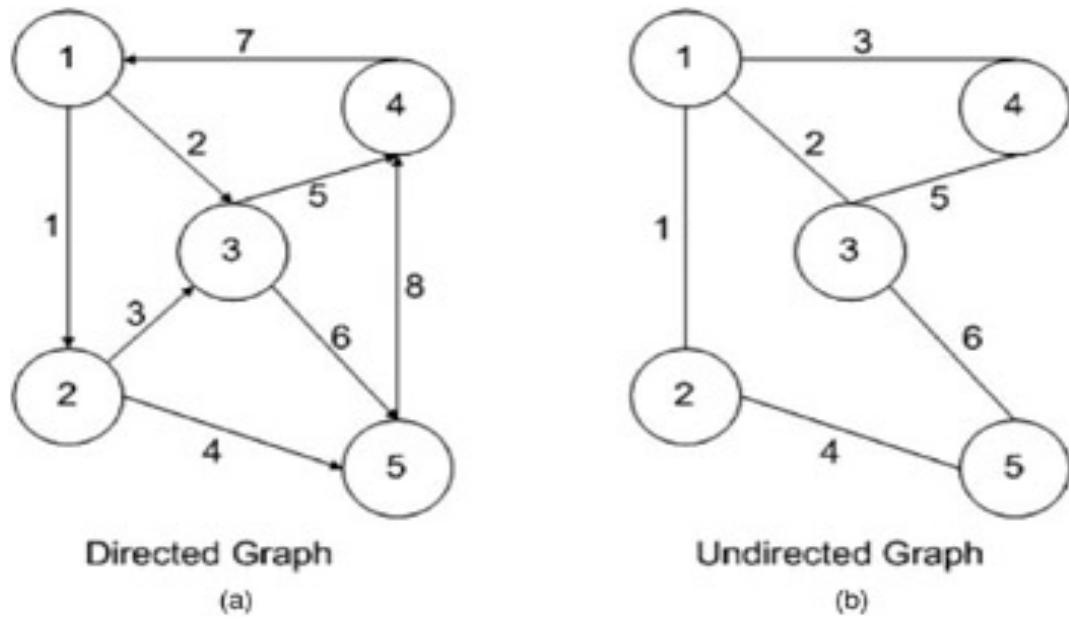
## 5. *Application Layer*

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. Salah satu kunci keberhasilan Android adalah banyaknya jumlah array (*vast array*) pada aplikasi yang dapat di install dari pihak ketiga, sehingga dapat memberikan kebebasan pada *user* untuk dapat melakukan interaksi dengan teman mereka melalui *social media*, edit film, *streaming* olah raga online secara *real time*, dan lain lainnya (Panigrahy, 2015).

## 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. 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,...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  $e = (u, v)$ .



**Gambar 2.5 Klasifikasi graf berdasarkan orientasinya;**

- (a) Graf terarah / *Directed graph*
  - (b) Graf tidak terarah / *Undirected graph*  
(Somani, 2005)

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. 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. 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 sebagai berikut:

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.

Salah satu kelemahan algoritma ini adalah algoritma ini tidak mampu mencari ujung/titik yang bernilai negatif. 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 (Triansyah, 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)**

<b>Procedure Dijkstra ( INPUT m : matriks dan a : simpul awal )</b>
<p>( Mencari lintasan terpendek dari simpul awal <b>a</b> ke semua simpul lainnya        Masukan : matriks ketetanggaan (<b>m</b>) dari graf berbobot <b>G</b> dan simpul awal <b>a</b>        Keluaran : lintasan terpendek dari <b>a</b> ke semua simpul lainnya )</p>
<p><u>Deklarasi</u></p> <ol style="list-style-type: none"> <li>1. <math>s_1, s_2, \dots, s_n</math> :integer (tabel integer)</li> <li>2. <math>d_1, d_2, \dots, d_n</math> :integer (tabel integer)</li> <li>3. <math>i, j, k</math>: integer</li> </ol>
<p><u>Algoritma ( langkah 0 / Inisialisasi)</u></p> <ol style="list-style-type: none"> <li>4. for <math>i \leftarrow 1</math> to <math>n</math> do</li> <li>5. <math>s_i \leftarrow 0</math></li> </ol>

6.	$di \leftarrow m_{ai}$
7.	Endfor
<u>Algoritma (langkah 1)</u>	
8.	Sa $\leftarrow 1$ (karena simpul a adalah simpul asal lintasan terpendek, jadi simpul a sudah pasti terpilih dalam lintasan terpendek )
9.	Daoo (tidak terdapat lintasan terpendek dari simpul a ke a)
<u>Algoritma (langkah 2, 3, ..., n-1:)</u>	
10.	For k $\leftarrow 2$ to n-1 do
11.	J $\leftarrow$ simpul dengan $sj = 0$ dan $dj$ minimal
12.	$Sj \leftarrow 1$ {simpul j sudah terpilih ke dalam lintasan terpendek}
13.	{perbarui tabel d}
14.	For semua simpul I dengan $si = 0$ do
15.	If $dj + mji < di$ then
16.	$Didj + mji$
17.	Endif
18.	Endfor
19.	Endfor

#### 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 di sebutkan 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  
(Saragih, 2014)**

Algoritma Floyd : Proses perhitungan <i>shortest path</i>
Floyd-Warshall(d,n)
<pre> 1. <b>for</b> k ←1 to n 2.     <b>for</b> j ←1 to n 3.         <b>for</b> i←1 to n 4.             <math>d_{ij}(k) = \min( d_{ij}(k-1), d_{ij}(k-1) )</math> 5.             <b>if</b> choose <math>d_{ij}</math> (k-1) + <math>d_{ij}(k-1)</math> <b>then</b> pred(ij) = k </pre>
Algoritma Floyd : Proses penentuan rute <i>shortest path</i>
Path (I, j)
<pre> 1. <b>If</b> pred(I, j) = nil <b>then</b> 2.             Return output(I, j) 3. <b>Else</b> 4.             Path(i, pred(I, j)) 5.             Path(pred(I, j), j) </pre>

#### 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, 2011)**

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

Dalam tabel ini, tabel 2.7, terlihat bahwa kompleksitas waktu yang dimiliki algoritma dijkstra memiliki kompleksitas yang lebih sederhana. 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)*

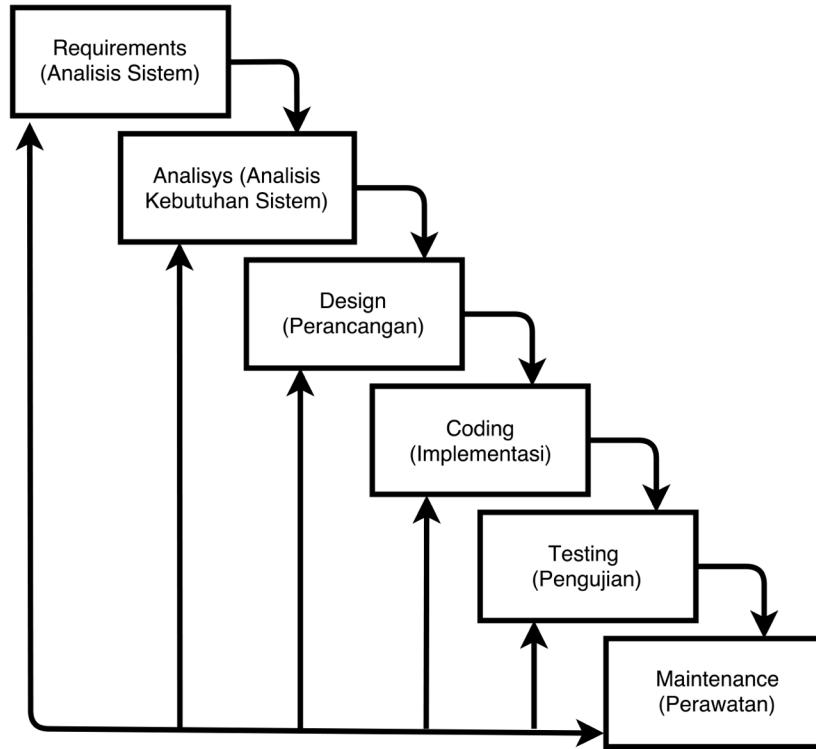
*System Development Life Cycle* atau SDLC atau Siklus Hidup Pengembangan Sistem adalah metode pengembangan sistem tradisional yang digunakan sebagian besar organisasi saat ini. *System Development Life Cycle* (SDLC) adalah kerangka kerja *framework* yang terstruktur yang berisi proses - proses sekuenzial 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). Banyak jenis model SDLC tradisional yang masing-masing memiliki kelebihan dan kekurangannya *Waterfall*, *V-Shape / Validation Process*, *2-1 Process (Incremental/Iterative)* dan jenis lainnya. (Khan, 2013).

Disebut model *waterfall* karena bentuknya yang bertingkat ke bawah dari satu fase ke fase lainnya. Model model *waterfall* ini biasanya disebut juga sebagai *classic life cycle*. Selanjutnya, penjelasan untuk model proses 2-1 (*2-1 Process Model*). *Incremental* adalah kombinasi dari desain berulang (metode *iterative*) dan *incremental building model* untuk pengembangan perangkat lunak. Model tersebut memiliki tujuh tahapan sebagai berikut: Perencanaan, persyaratan, analisis, pelaksanaan, penyebaran, pengujian, dan evaluasi. Perbandingan dari beberapa model – model SDLC dapat terlihat pada tabel 2.8.

**Tabel 2.8 Perbandingan model – model SDLC  
(Waterfall, 2-1 Process dan V-Process) (Khan, 2013)**

No	Kriteria Evaluasi	Model SDLC Tradisional		
		Waterfall	2-1 Process Incremental/ Iterative	V-Proces
1	Kritikalitas Bisnis <i>(Business Criticality)</i>	Low	Medium	High
2	Keterlibatan Customer	Low	Medium	Low
3	Kejelasan Kebutuhan <i>(Requirements Clarity)</i>	High	Medium	High
4	Kebutuhan yang berubah-ubah <i>(Requirements Volatiliy)</i>	Low	High	Low
5	Ukuran Proyek <i>(Project Size)</i>	Small	Large	Large
6	Kompleksitas Bisnis	Low	High	Low
7	Kompleksitas Teknis	Low	High	Medium

Atas perbandingan yang dilakukan pada tabel 2.8, maka dapat disimpulkan bahwa model yang tepat yang akan diimplementasikan pada aplikasi OAM adalah model *waterfall*. Tahapan-tahapan model ini diantaranya adalah Requirements (Analisis Sistem), Analysis (Analisis Kebutuhan Sistem), Design (Perancangan), Coding (Implementasi), Testing (Pengujian) dan Maintenance (Perawatan), lihat gambar 2.6 (Pressman, 1997). Penjelasan metode *System Development Life Cycle* (SDLC) adalah sebagai berikut:



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

### 1) Requirements (Analisis Sistem)

Tahap 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 dapat di lihat di lingkungan masyarakat, maka *developer* akan melakukan *brainstorming* untuk meningkatkan performa dari aplikasi tersebut. 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) *Analysis* (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. Untuk mendapatkan data analisis pada tahap ini akan dilakukan elisitasi kebutuhan (*requirement elicitation*). Elitisasi kebutuhan (*requirement elicitation*) merupakan kumpulan aktivitas yang dilakukan bertujuan untuk menemukan kebutuhan – kebutuhan suatu sistem melalui komunikasi dengan pelanggan, pengguna sistem dan pihak lain yang memiliki peran penting dalam pengembangan sistem aplikasi (Sommerville, 1997). Elitisasi kebutuhan dapat dilihat pada lampiran 1.

## 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 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*. 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 terdapat fungsi-fungsi yang diujikan telah memenuhi spesifikasi kebutuhan

yang telah di buat 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* 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 di sertai 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 (*analysis*).

##### 2) *Analysis* (Analisis Kebutuhan Sistem)

Tahap kedua dari SDLC merupakan analisis kebutuhan sistem (*analysis*). Tahap ini merupakan tahap analisis kebutuhan sistem untuk membangun aplikasi OAM. Pada tahap analisis ini terdapat 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

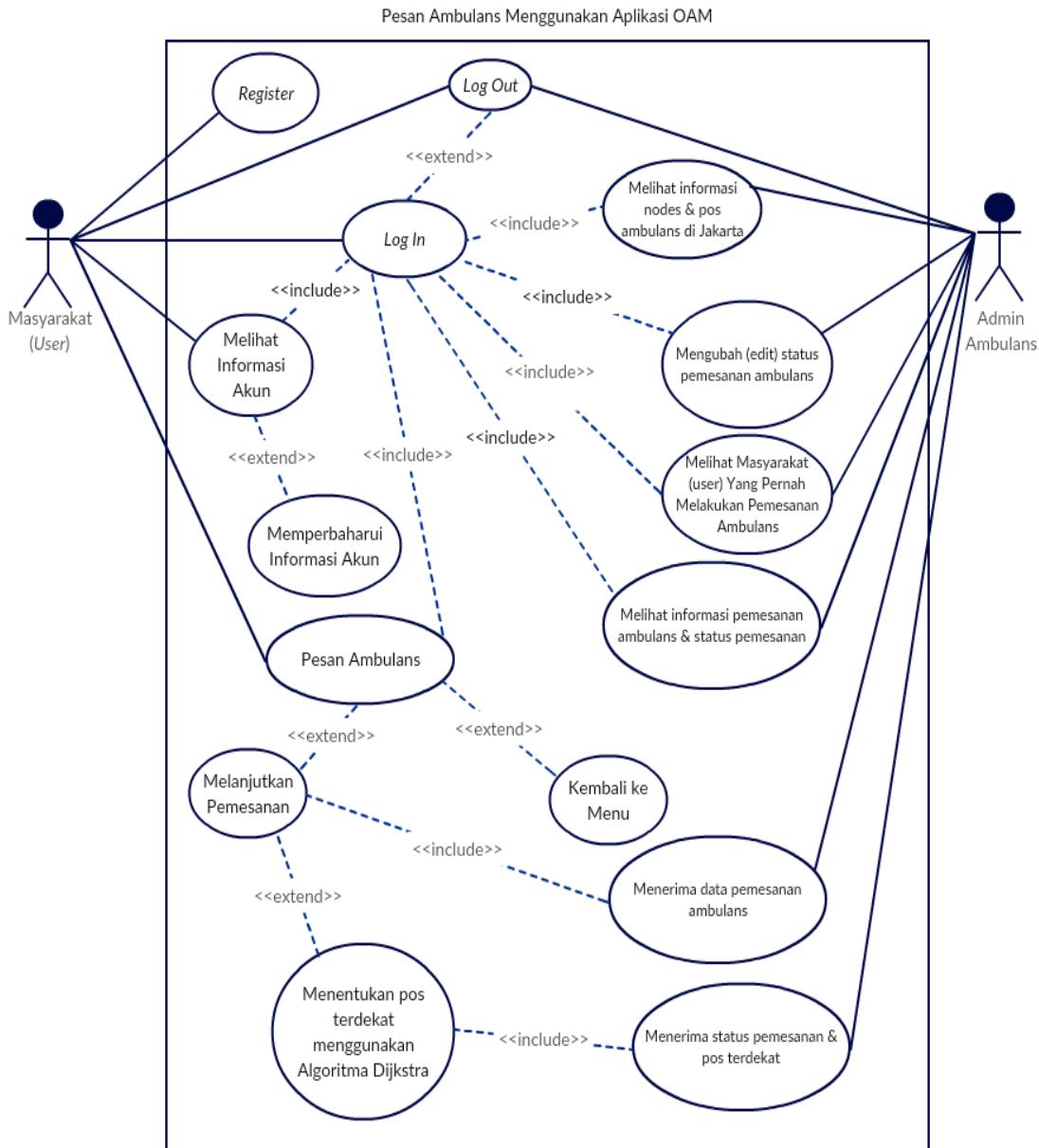
Kebutuhan perangkat lunak adalah perangkat lunak yang dibutuhkan dalam pembangunan sistem aplikasi ini, diantaranya sebagai berikut

1. Android Studio 2.2.3
2. Adobe Photoshop CS6
3. Adobe Illustration CS6
4. Adobe Experience Design CC 0.6.8.6 (Beta)
5. MongoChef Core 4.5.2
6. Gmap3
7. Web Browser : Chrome (55.0.2883.95) & Safari 9.0.1 (11601.2.7.2)
8. Twitter Bootstrap 3.2.1
9. LoopBack 2.4.8
10. NodeJS 7.2.1
11. ExpressJS 4
12. CoreUI Website Template

### 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*.

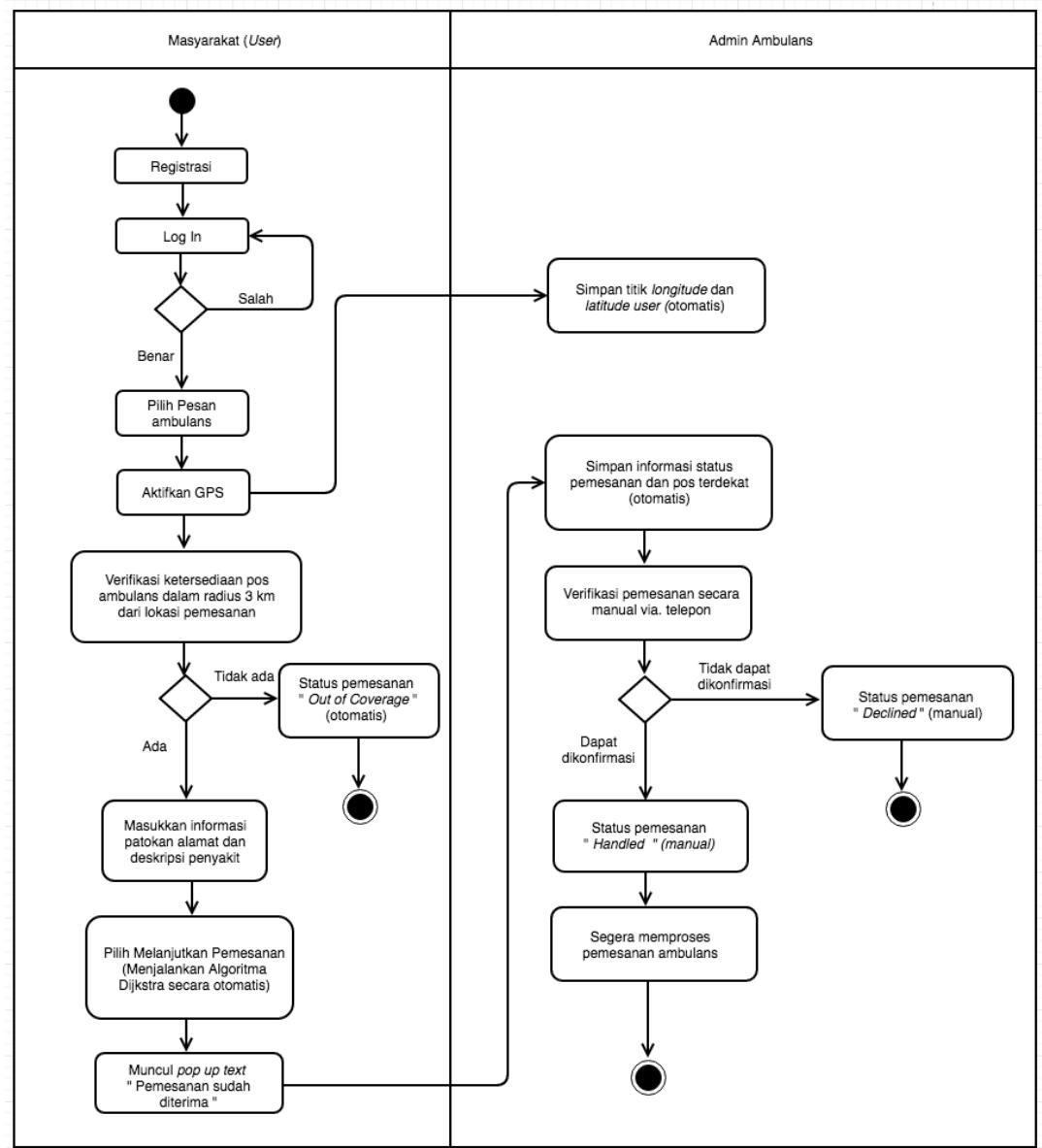
a. *Use case diagram*



**Gambar 3.1 Use Case Diagram Aplikasi OAM**

*Use case* diagram yang digunakan pada aplikasi OAM dapat dilihat pada gambar 3.1. Deskripsi *use case diagram* aplikasi OAM (gambar 3.1) dapat dilihat pada lampiran 1.

b. *Activity Diagram* (Diagram Aktivitas)



Gambar 3.2 *Activity Diagram* Aplikasi OAM

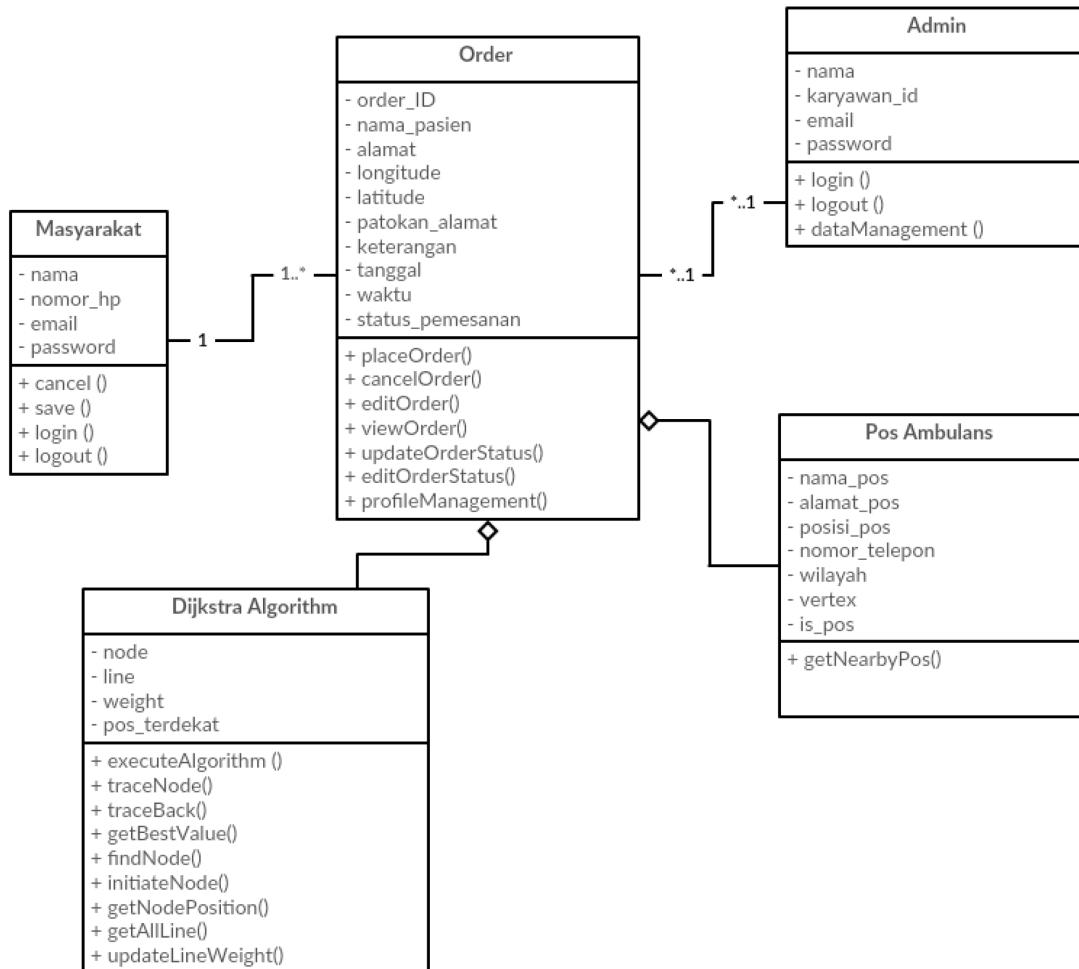
*Activity diagram* (diagram aktivitas) yang digunakan pada dapat aplikasi OAM terlihat pada gambar 3.2. Pada gambar 3.2, dapat dilihat bahwa diagram

aktivitas tersebut merepresentasikan alur pemesanan ambulans menggunakan aplikasi OAM. Diagram tersebut menggambarkan proses interaksi antara masyarakat sebagai *user* dengan admin ambulans. Proses pemesanan ambulans dimulai dari masyarakat (*user*) melakukan registrasi. Setelah melakukan registrasi masyarakat (*user*) harus *log in* terlebih dahulu untuk dapat masuk ke fitur utama pada aplikasi. Setelah masuk ke dalam aplikasi melalui *log in*, *user* dapat melakukan pemesanan ambulans dengan memilih tombol “pesan ambulans”. *Smartphone user* akan memunculkan *pop up box* untuk meminta izin menghidupkan koneksi GPS apabila sebelumnya belum dihidupkan. Sistem di Android akan mengirimkan informasi titik *longitude* dan *latitude* secara otomatis kepada *webservice* admin ambulans. Verifikasi akan dilakukan untuk mengetahui apakah pos ambulans berada dalam radius 3km dari lokasi pemesanan, apabila terdapat *user* akan di minta untuk mengisi beberapa informasi tambahan berupa patokan alamat dan deskripsi penyakit. Apabila tidak ditemukan pos ambulans dalam radius 3km dari lokasi pemesanan, maka pemesanan tidak akan dilanjutkan dan diubah status pemesanannya secara otomatis “*out of coverage*”. Setelah mengisi beberapa informasi tambahan, *user* akan memilih tombol “lanjutkan pemesanan” dan sistem akan melanjutkan pemesanan tersebut dengan menjalankan algoritma Dijkstra. Setelah aplikasi berjalan menghitung pos terdekat, aplikasi akan memunculkan *pop up box* dengan tulisan “pemesanan sudah diterima”.

Selanjutnya sistem akan menyimpan status pemesanan dan pos terdekat hasil dari perhitungan algoritma Dijkstra sebelumnya. Verifikasi akan dilakukan kembali untuk mengkonfirmasi kebenaran pemesanan secara manual via telepon. Apabila pemesanan tidak dapat dikonfirmasi maka pemesanan akan diubah ditolak dan statusnya di ubah menjadi “*declined*”. Apabila pemesanan dapat dikonfirmasi melalui via telepon, admin ambulans akan segera

memproses pemesanan ambulans dengan mempersiapkannya dan mengarahkannya ke tempat tujuan.

### c. Class Diagram



Gambar 3.3 *Class Diagram* Aplikasi OAM

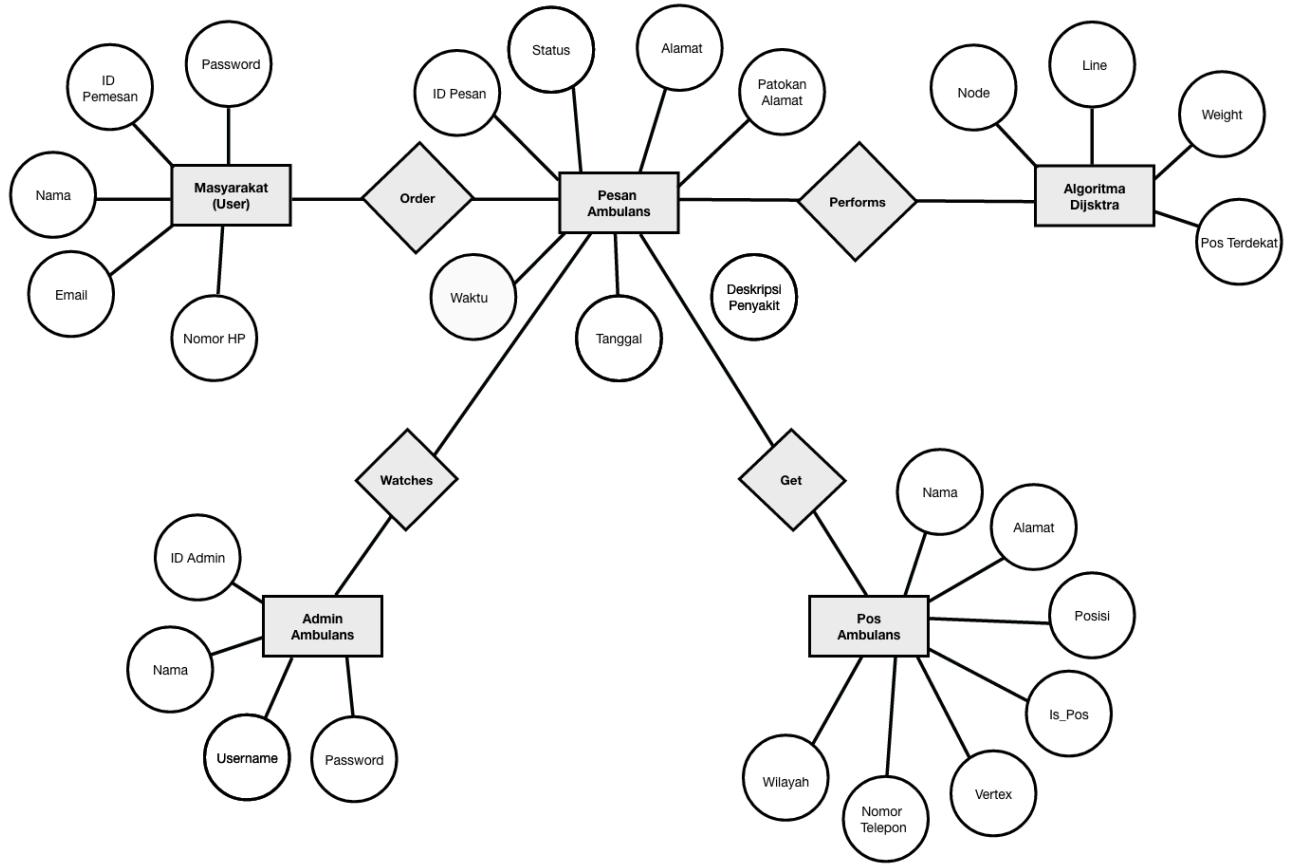
*Class Diagram* merupakan diagram yang menggambarkan sudut sistem dari struktur dan definisi *class* serta hubungannya (*relationship*). *Class* memiliki 3 area pokok, yaitu nama, atribut, dan operasi. Nama berfungsi untuk memberikan identitas pada sebuah *class*. Atribut berfungsi untuk memberikan

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 order*, *class log in admin*, *class pos ambulans*, dan *class Dijkstra algorithm*. *Class masyarakat* memiliki beberapa atribut seperti nama, nomor hp, email, dan *password*. *Class masyarakat* memiliki hubungan asosiasi berarah (*directed association*) dengan *class pesan ambulans*. *Class order* memiliki beberapa atribut seperti *order id*, nama pasien, alamat, titik *latitude & longitude*, patokan alamat, keterangan keadaan, tanggal, waktu, dan status pemesanannya. *Class order* memiliki hubungan dengan *class admin*. *Class admin* memiliki beberapa atribut berupa nama admin, id karyawan, *email* dan *password*. *Class Diagram* yang digunakan pada aplikasi *order ambulance mobile* (OAM) dapat dilihat pada gambar 3.3.

#### d. Database Design

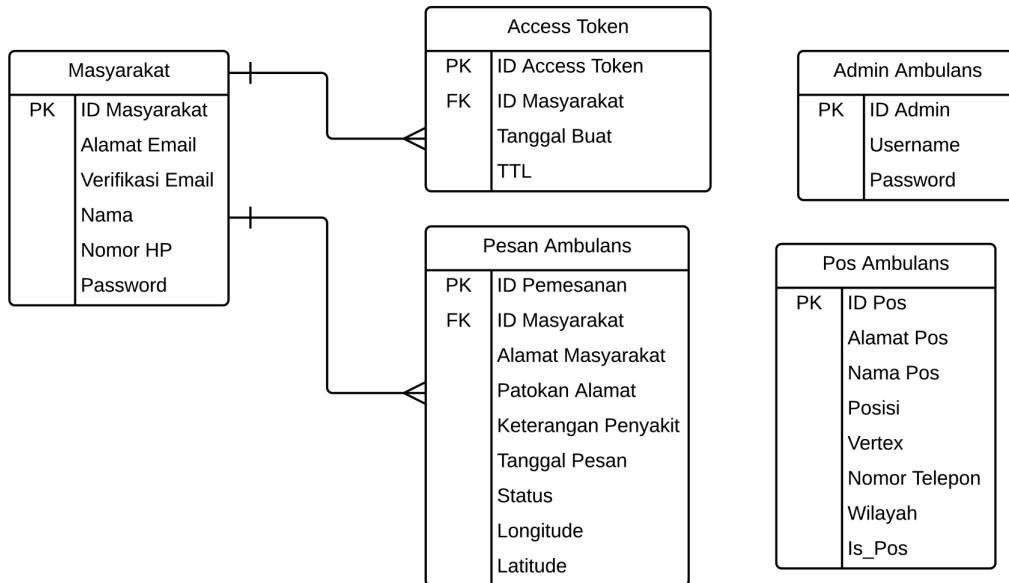
Sistem ini memiliki dua sisi yaitu di sisi *Server* dan sisi *Client*. Di sisi *Server* ini, yang merupakan *Web Service* aplikasi, digunakan oleh pihak admin ambulans untuk mengolah informasi oleh masyarakat yang telah melakukan pemesanan ambulans melalui aplikasi OAM. Sementara sisi *Client* merupakan aplikasi berbasis android yang digunakan oleh masyarakat sebagai pengguna (*user*) untuk melakukan pemensanan ambulans. Pada gambar 3.4 dapat dilihat bahwa sistem ini memiliki 5 tabel yang diantaranya adalah Tabel Masyarakat, Tabel Pesan Ambulans, Tabel Admin Ambulans, Tabel Pos Ambulans dan Tabel Algoritma Dijkstra



Gambar 3.4 Entity Relationship Diagram (ERD) pada OAM

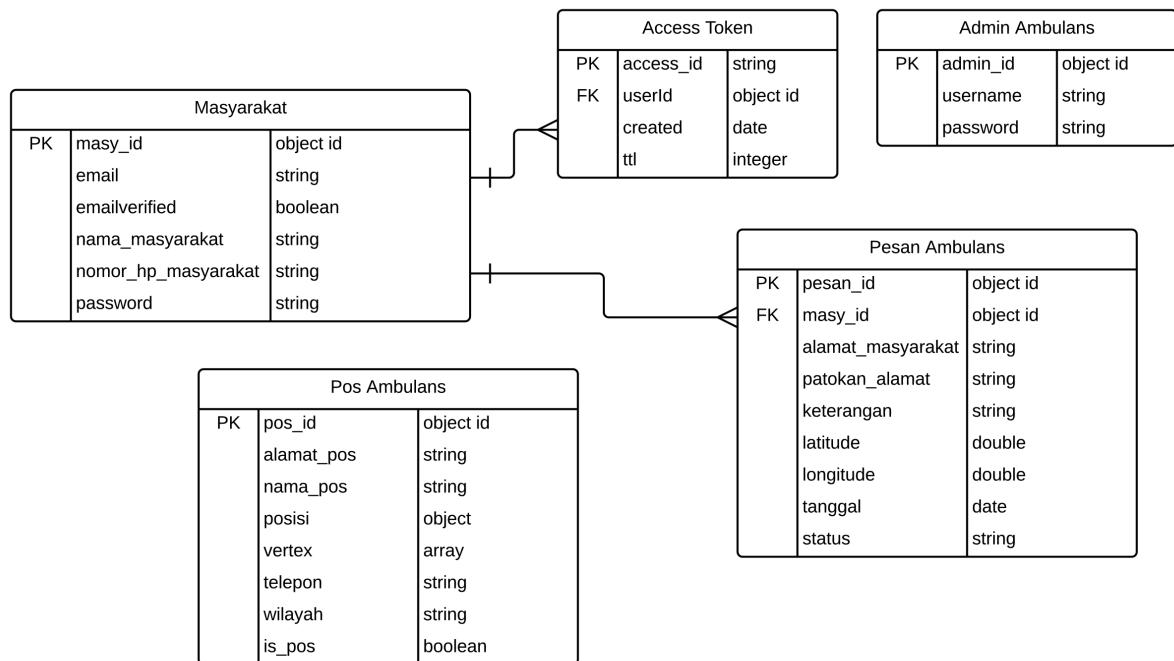
Ketiga tabel ini masing masing memiliki attribut dan memiliki hubungan antara satu tabel dengan tabel lainnya (*relationship*). *Logical database model* adalah suatu model informasi yang digunakan pada organisasi, instansi, maupun perusahaan berdasarkan pada model data yang spesifik namun tidak tergantung pada *Database Management System* (DBMS) yang khusus dan memiliki pertimbangan fisik lainnya (Connolly, 2010). 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 3.5 Logical Database Model pada aplikasi OAM**

Tidak banyak perbedaan antara *physical design* dengan *logical design*, hanya saja masing-masing atribut pada tiap tabel telah memiliki tipe data pada *physical design*. Teridentifikasinya daftar tabel, atribut, serta tipe data atribut pada tiap tabel menandakan bahwa rancangan basis data sudah selesai dilakukan. Rancangan ini kemudian akan diimplementasikan untuk mengorganisir penyimpanan data yang dibutuhkan oleh sistem nantinya. Gambar 3.6 menunjukkan *physical database model* yang akan diimplementasikan pada aplikasi OAM.



**Gambar 3.6 Physical Database Model pada aplikasi OAM**

#### e. User Interface Design aplikasi OAM

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

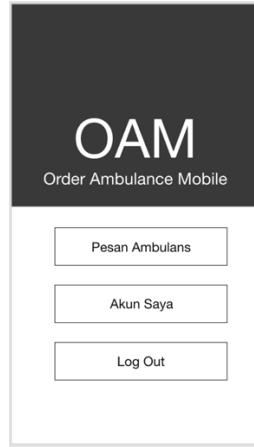
The wireframe shows a registration form. At the top, there are two buttons: 'Log In' on the left and 'Register' on the right, with a horizontal line underneath. Below these are four input fields: 'Alamat Email', 'Password', 'Nama Lengkap', and 'Nomor Handphone', each preceded by a short label and a horizontal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

**Gambar 3.7 Perancangan Antarmuka pada Aplikasi OAM Bagian Satu**

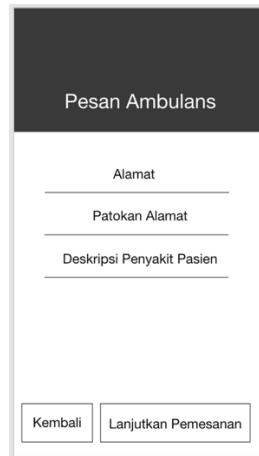
The wireframe shows a login form. At the top, there are two buttons: 'Log In' on the left and 'Register' on the right, with a horizontal line underneath. Below these are two input fields: 'Alamat Email' and 'Password', each preceded by a short label and a horizontal line. At the bottom is a large rectangular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



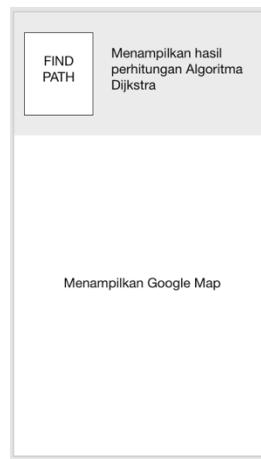
**Gambar 3.9 Perancangan Antarmuka pada Aplikasi OAM Bagian Tiga**



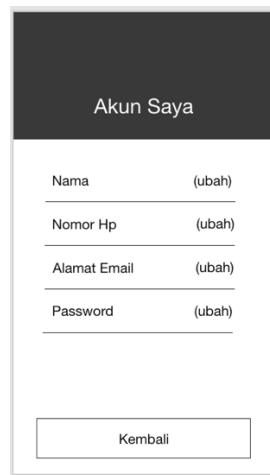
**Gambar 3.10 Perancangan Antarmuka pada Aplikasi OAM Bagian Empat**

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 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 testing* dan metode *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 apakah terdapat yang kesalahan 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 terdapat sesuai batasan-batasannya.
3. Pengujian pada struktur data yang sifatnya internal dan yang terjamin validitasnya.

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 pengujian aplikasi OAM metode menggunakan metode *black box testing* dan metode *white box testing* dalam penelitian ini dapat dilihat 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* 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 melakukan pemesanan ambulans di wilayah Jakarta.

## BAB IV

### IMPLEMENTASI DAN PENGUJIAN APLIKASI

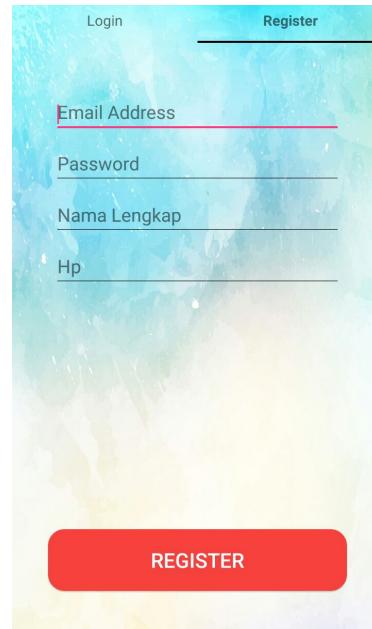
#### 4.1 Implementasi

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*.

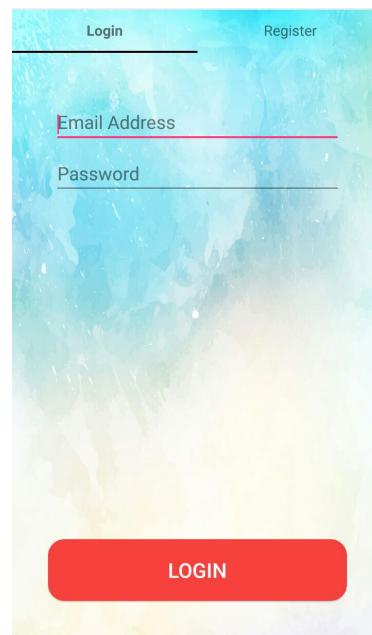
##### 4.1.1 Implementasi Aplikasi OAM pada Android

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

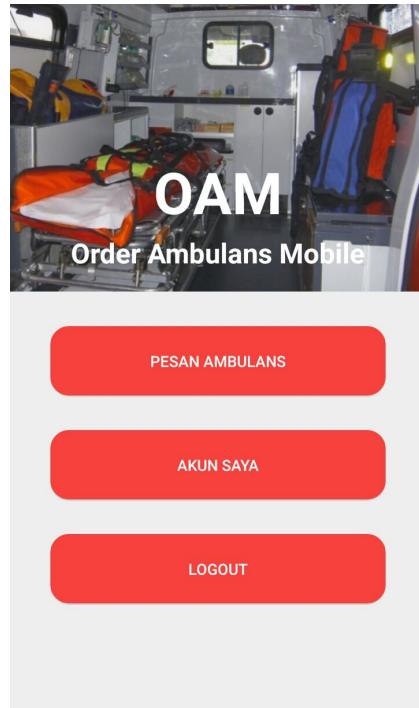
yang telah melakukan registrasi. Untuk melakukan *log in* dibutuhkan alamat email dan password yang pernah didaftarkan sebelumnya.



**Gambar 4.1 Tampilan Antarmuka Android Halaman Register**



**Gambar 4.2 Tampilan Antarmuka Android Halaman Log In**



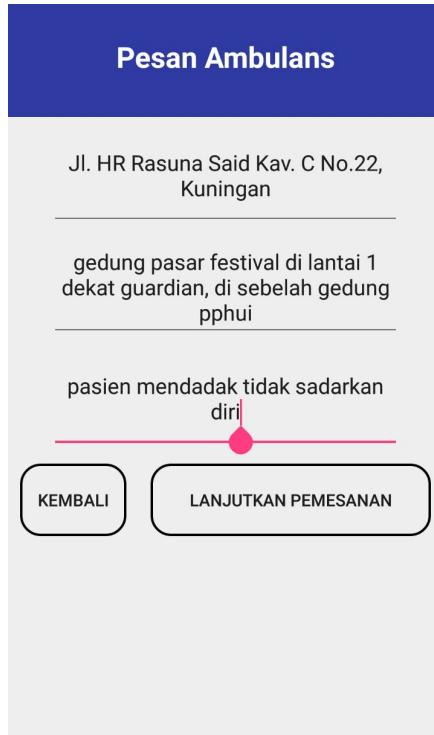
**Gambar 4.3 Tampilan Antarmuka Android Halaman Menu**

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 diisikan sebagai contoh pengisian pada kolom alamat adalah Jl. HR Rasuna Said Kav. C No.22, Kuningan yang didapatkan secara otomatis dari aplikasi OAM. Informasi yang diisikan 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 diisikan informasi di aplikasi bahwa pasien mendadak tidak sadarkan diri.

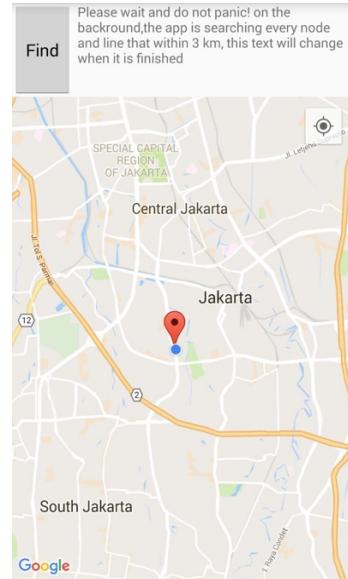


**Gambar 4.4 Tampilan Antarmuka Android Halaman Pesan Ambulans**

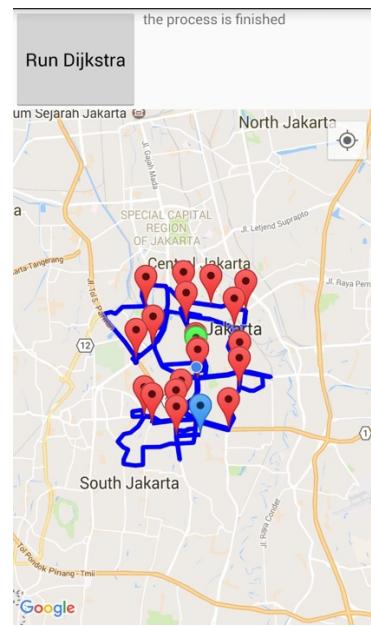


**Gambar 4.5 Tampilan Antarmuka Android Halaman Pesan Ambulans  
(filled)**

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 *smartphonennya*, maka aplikasi pemesanan ambulans OAM tidak dapat dijalankan karena kedua hal tersebut dibutuhkan untuk memesan ambulans dan aplikasi tetap berada di menu utama. Setelah menekan tombol “lanjutkan pemesanan” maka pengguna (*user*) akan dibawa ke halaman pemesanan selanjutnya, lihat gambar 4.6. Akan terdapat kalimat yang meminta pengguna menunggu sesaat ketika aplikasi sedang berjalan. Apabila aplikasi telah mendapatkan informasi yang dibutuhkan seperti titik lokasi pemesan, rute dan pos terdekat, maka tampilan akan berubah menjadi seperti pada gambar 4.7.

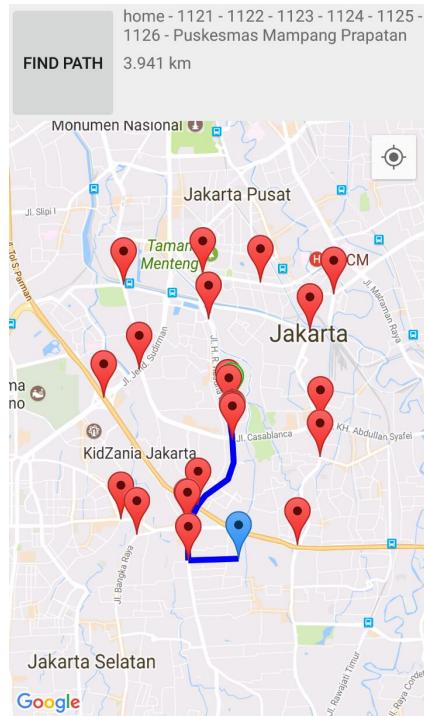


**Gambar 4.6 Tampilan Antarmuka Android Halaman Pesan Ambulans dengan Algoritma Dijkstra**



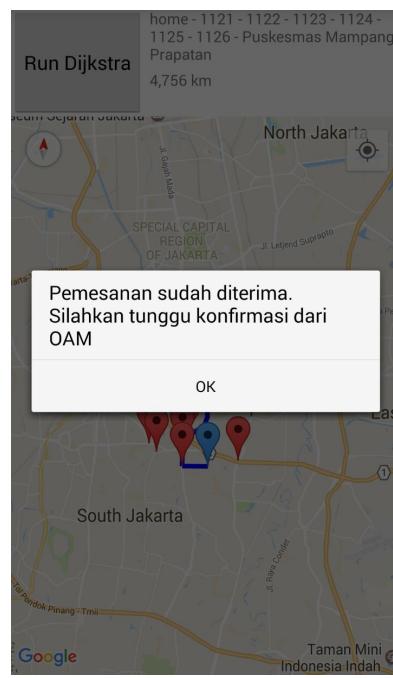
**Gambar 4.7 Tampilan Antarmuka Android Halaman Pesan Ambulans dengan status “finished”**

Gambar 4.7 Merupakan halaman yang akan ditampilkan ketika pengguna menekan tombol “Find” pada gambar 4.6 sebelumnya. Halaman yang terdapat pada gambar 4.7 menunjukkan titik – titik rute (nodes) , titik lokasi pemesan, dan titik pos yang tersedia dalam radius 3 km. 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 4.7, setelah pengguna menekan tombol / button “find path” maka akan muncul tampilan aplikasi seperti 4.8. Dalam proses pemesanan ini, sudah terjadi pengiriman informasi dari aplikasi OAM Android ke *web service* OAM.



**Gambar 4.8 Tampilan Antarmuka Android Halaman Pesan Ambulans dengan jalur hasil perhitungan**

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 4.8, maka aplikasi akan segera memunculkan tampilan hasil aplikasi seperti pada gambar 4.9. Dalam gambar 4.9 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 4.9 terlihat rute terdekat yang dijadikan sebagai hasil algoritma di dalam aplikasi. 1121, 1122, 1123, 1124, 1125, 1126 merupakan nama titik-titik rute yang telah terdaftar pada aplikasi OAM. Pencarian algoritma ini telah menemukan bahwa pos ambulans terdekat terletak di Puskesmas Mampang Prapatan dengan jarak 4,7 km.

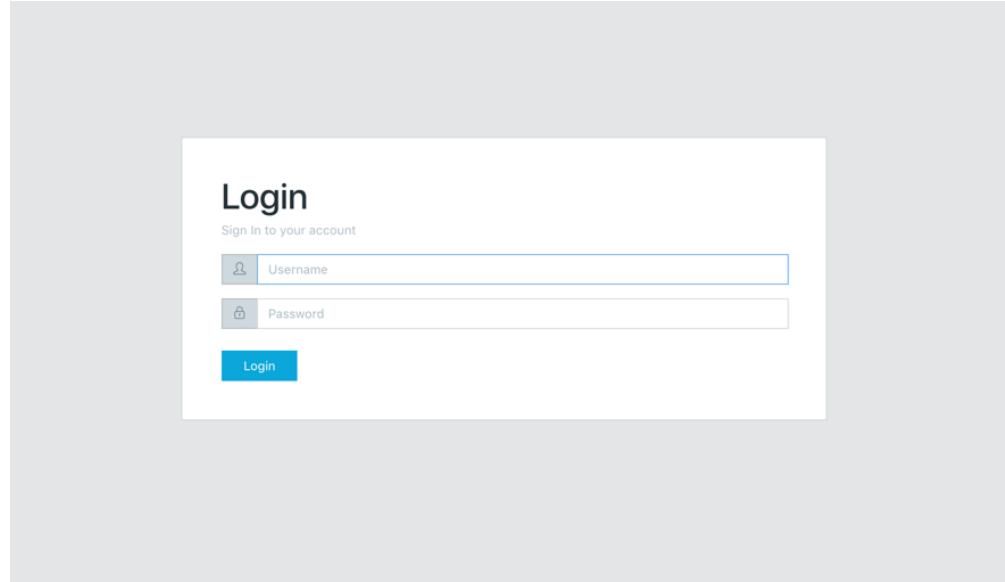


**Gambar 4.9 Tampilan Antarmuka Android Halaman Pesan Ambulans telah terkonfirmasi**

*Pop up box* bertuliskan “Pemesanan sudah diterima. Silahkan tunggu konfirmasi dari OAM” akan muncul ketika proses pemesanan ambulans melalui aplikasi telah selesai dan berhasil dilakukan seperti yang ditunjukkan pada gambar 4.9. 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.

#### 4.1.2 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 4.10. 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 4.11.



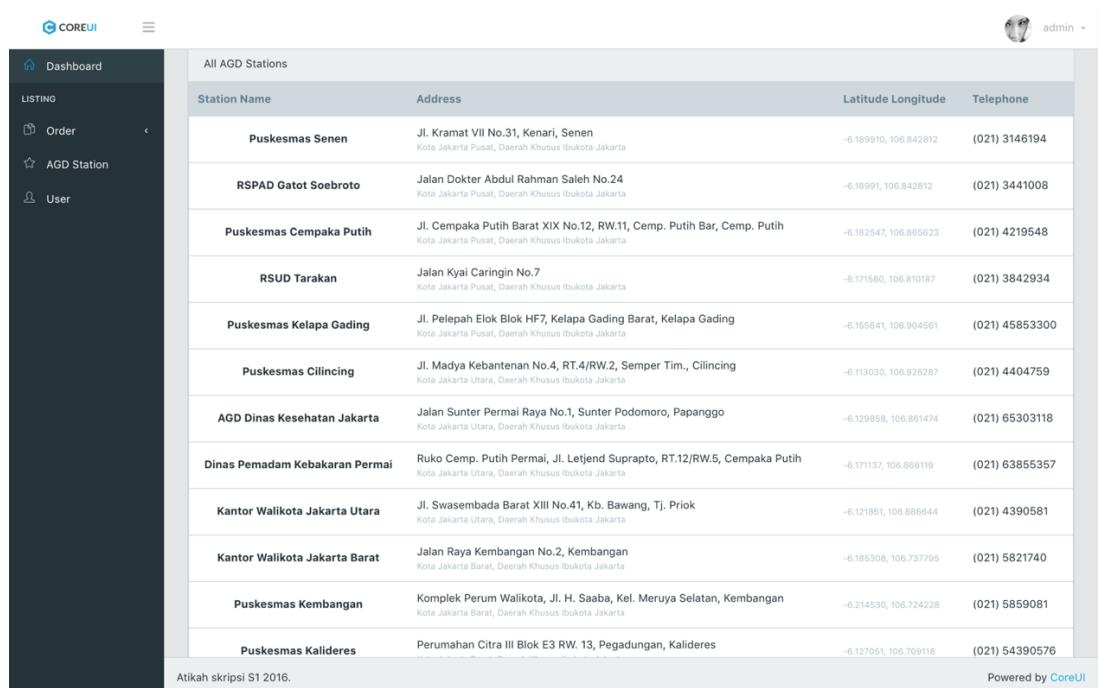
Gambar 4.10 Tampilan Antarmuka Website OAM Halaman Log In Admin

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	-6.189910, 106.842812	(021) 3441008

Gambar 4.11 Tampilan Antarmuka Website OAM Halaman Main Menu

Pada halaman *main menu* website OAM di gambar 4.11, 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. Data yang terlihat pada 4.11 merupakan data yang didapatkan oleh penulis lalu dimasukkan (*input*) secara manual sehingga dapat ditampilkan dengan baik di *website* admin ambulans. 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 terdapat pada AGD 118 di Jakarta. Tampilan informasi pos – pos ambulans dapat dilihat pada gambar 4.12.



The screenshot shows a web application interface for managing Ambulans Gudang Digital (OAM). On the left, there is a sidebar menu with options: Dashboard, LISTING, Order, AGD Station, and User. The main content area is titled "All AGD Stations" and displays a table with the following data:

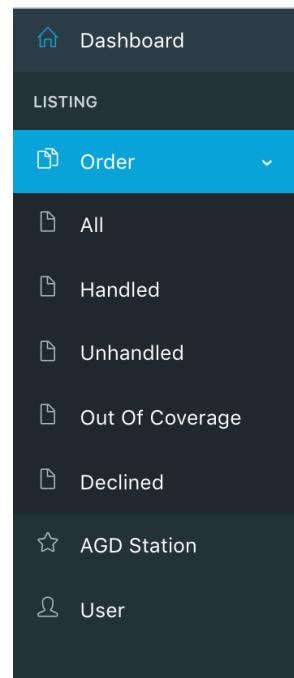
Station Name	Address	Latitude	Longitude	Telephone
Puskesmas Senen	Jl. Kramat VII No.31, Kenari, Senen Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.189910	106.842812	(021) 3146194
RSPAD Gatot Soebroto	Jalan Dokter Abdul Rahman Saleh No.24 Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.18991	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. Peleleh Elo Blok HF7, Kelapa Gading Barat, Kelapa Gading Kota Jakarta Pusat, Daerah Khusus Ibukota Jakarta	-6.155641	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. Bawang, Tj. Priok Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta	-6.121851	106.886644	(021) 4390581
Kantor Walikota Jakarta Barat	Jalan Raya Kembangan No.2, Kembangan 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

At the bottom of the table, it says "Atikah skripsi S1 2016." On the right side, there is a "Powered by CoreUI" logo.

**Gambar 4.12 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 4.13. Tombol – Tombol tersebut merupakan jenis status yang terdapat pada proses pemesanan ambulans.



Gambar 4.13 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 seluruh status dapat dilihat pada gambar 4.14. 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 4.15. Pada halaman tersebut akan terlihat informasi pos – pos ambulans yang terdapat pada AGD 118 di Jakarta.*

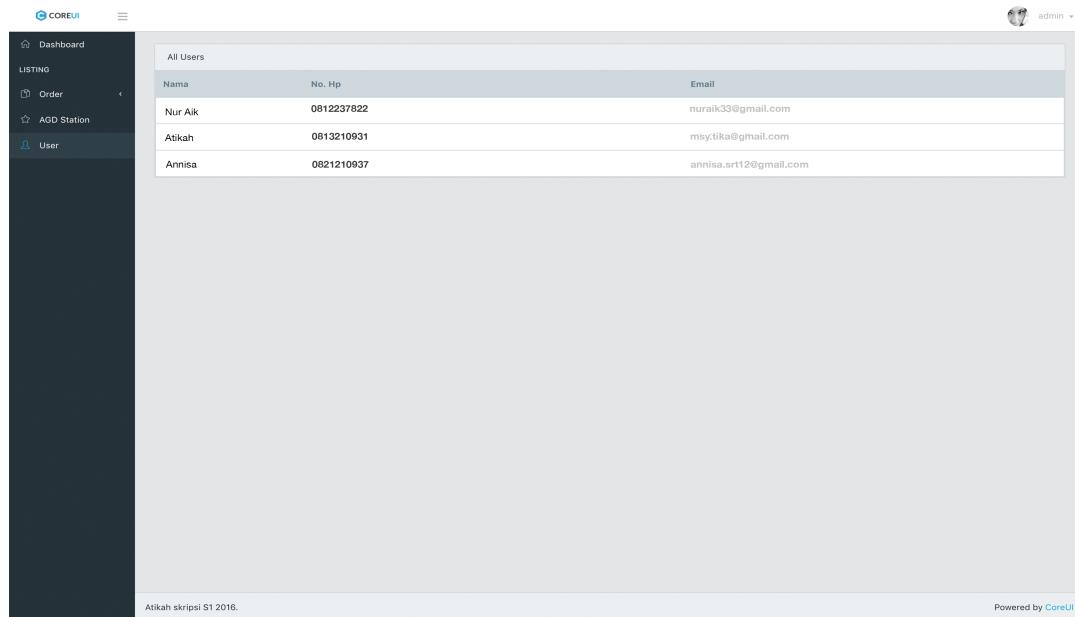
Date/Time	Address	Disease	Name	Phone	Status	Set Status	Pos Terdekat
2017-01-11 16:27:26	Jalan Haji R. Rasuna Said No.Kav XG/6-7 Karet Kuningan Kecamatan Setiabudi Kota Jakarta Selatan Daerah Khusus Ibukota Jakarta 12940	pingsan	atikah	0811768949	unhandled	<a href="#">Handle Order</a> <a href="#">Decline Order</a>	Puskesmas Mampang Prapatan
2017-01-11 16:25:35	Jalan Haji R. Rasuna Said No.Kav XG/4-5 Karet Kuningan Kecamatan Setiabudi Kota Jakarta Selatan Daerah Khusus Ibukota Jakarta 12940	Kecelakaan	atikah	0811768949	unhandled	<a href="#">Handle Order</a> <a href="#">Decline Order</a>	Puskesmas Mampang Prapatan

**Gambar 4.14 Tampilan Antarmuka Website OAM Halaman Pemesanan Ambulans Seluruh Status**

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	-6.18891	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	106.869623	(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 Blok HF7, Kelapa Gading Barat, Kelapa Gading Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta	-6.155641	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	-6.129658	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. Kawung, 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 Kembaran	Kompleks Perum Walkota, Jl. H. Saaba, Kel. Meruya Selatan, Kembaran Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta	-6.124530	106.724228	(021) 5859081
Puskesmas Kalideres	Perumahan Citra III Blok E3 RW. 13, Pegadungan, Kalideres Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta	-6.127051	106.709518	(021) 54390576
Puskesmas Grogol Petamburan	Kompleks Tamans Duta Mas, Jl. Wijaya Kusuma 3 Blok. F No. 1 Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta	-6.149581	106.776740	(021) 5648379
Puskesmas Cenkareng	Jalan Kamal Raya No. 02, Cengkareng Bar., Cengkareng Kota Jakarta Barat, Daerah Khusus Ibukota Jakarta	-6.144174	106.728240	(021) 29038167

**Gambar 4.15 Tampilan Antarmuka Website OAM Halaman AGD Stations**

Informasi yang ditampilkan sama dengan yang ditampilkan pada menu utama website di gambar 4.12 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 4.16. 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 sidebar with a dark background containing navigation links: Dashboard, LISTING, Order, AGD Station, and User. The User link is highlighted with a blue background. The main content area has a light gray background. At the top right, there is a user profile icon and the text "admin". Below the sidebar, a table titled "All Users" displays three rows of data:

Nama	No. Hp	Email
Nur Aik	0812237822	nuraiik33@gmail.com
Atikah	0813210931	misy.tika@gmail.com
Annisa	0821210937	annisa.srt12@gmail.com

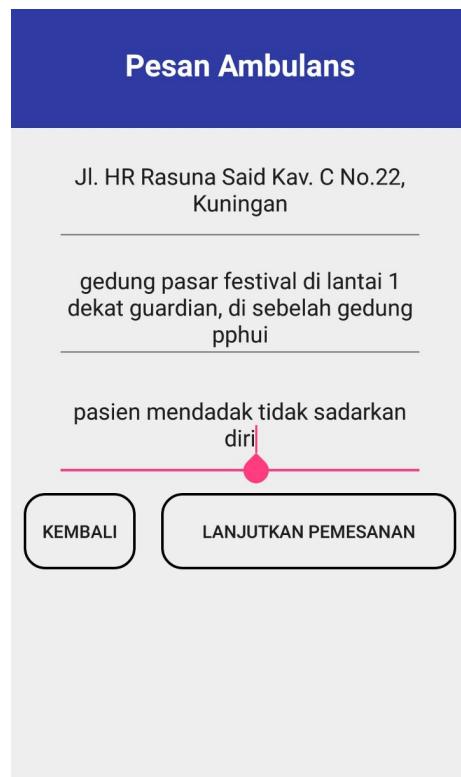
At the bottom of the page, there is a footer bar with the text "Atikah skripsi S1 2016." on the left and "Powered by CoreUI" on the right.

**Gambar 4.16 Tampilan Antarmuka Website OAM Users**

## 4.2 Pengujian Aplikasi OAM

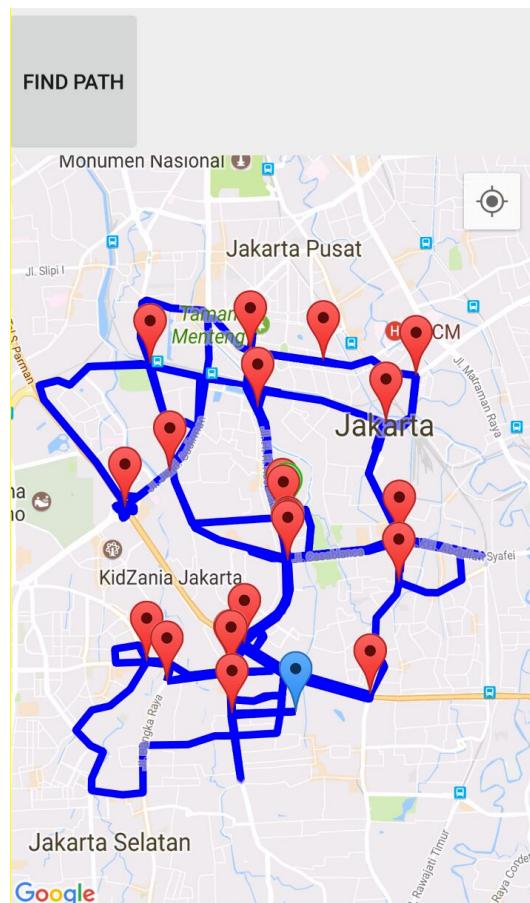
Tahap ini merupakan tahap pengujian (*testing*) karena pengujian aplikasi OAM akan dilakukan pada bab ini. Seperti yang dibahas pada BAB II, Algoritma Dijkstra yang digunakan pada aplikasi OAM merupakan algoritma pencarian jarak terpendek (*shortest path algorithm*) merupakan solusi untuk mencari jarak 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 4.17 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 4.17. Aplikasi akan merespon tindakan tersebut dengan menampilkan halaman peta dan pencarian pos terdekat. Tampilan selanjutnya terlihat pada gambar 4.18. 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 4.18 Tampilan Antarmuka Android Halaman Pesan Ambulans**

Pada saat pengguna (*user*) klik “lanjutkan pemesanan” pada halaman pengisian informasi pemesanan di gambar 4.17, 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. *Source code* aplikasi yang ditampilkan pada Android Studio terlihat pada gambar 4.19.

```

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 4.19 Sebagian *source code* pada Android Studio**

*Source code* pada gambar 4.19 tersebut menjelaskan bahwa aplikasi hanya akan menampilkan pos yang berada dalam radius 3 km di sekitar pemesan (*user*) ambulans, hal tersebut didefinisikan pada garis ini “app.models.PosAmbulans.find

`({where:{posisi:{near:userLocation,maxDistance: 3, unit: 'kilometers'}}})`. *Source code* untuk inisialisasi titik-titik rute jalan (*nodes*) dapat dilihat pada gambar 4.20.

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

**Gambar 4.20 Sebagian *source code* 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 *source code* method *initiateNode* dapat dilihat pada gambar 4.21.

```
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 4.21 Sebagian *source code* pada Android Studio Tiga**

Algoritma Dijkstra pada aplikasi OAM secara keseluruhan digambarkan pada *source code* Android yang terlihat pada gambar 4.22. 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 4.22 Sebagian *source code* 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 4.23. 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());

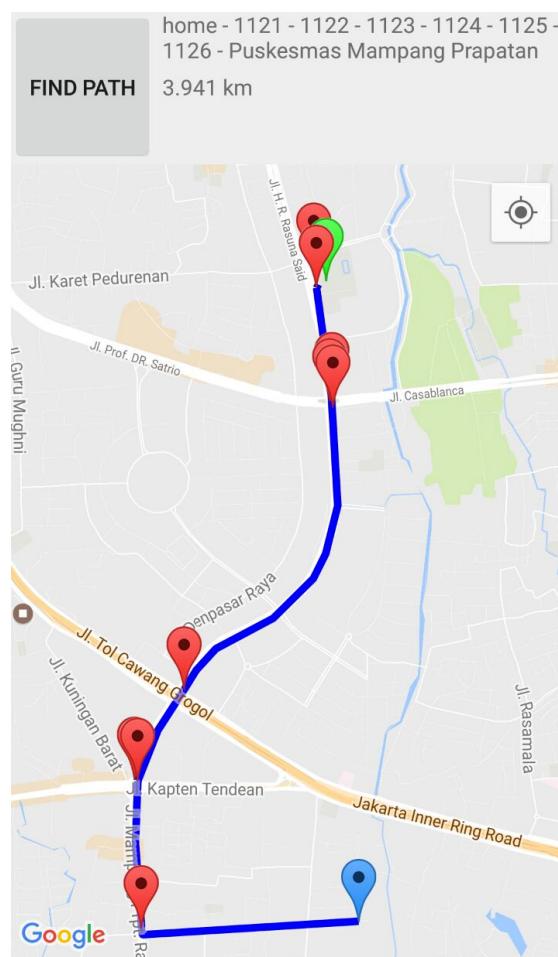
        if (i == activeNode.getVertexLength() - 1) {
            activeNode.setActive(false);
            getBestValue();
            strActiveName = strBestNameLinear;
            initialWeight = intBestWeightLinear;
        }
    }
}

```

**Gambar 4.23 Sebagian *source code* pada Android Studio Lima**

Kemudian dari node aktif tersebut di ambil / *collect* list-list tetangga 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 4.24. 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 3,941 km dari titik lokasi pemesan (*user*).



Gambar 4.24 Tampilan Antarmuka Android Pesan Ambulans

#### **4.2.1 *White Box Testing***

Pengujian *white box testing* yang dilakukan dalam penelitian ini sesuai dengan tahap metode penelitian SDLC yang di rancang pada BAB III. Pengujian menggunakan *white box* diterapkan pada beberapa fungsi Algoritma Dijkstra. Hasil dari pengujian *white box* terhadap beberapa fungsi Algoritma Dijkstra pada aplikasi OAM dapat dilihat di lampiran 2.

#### **4.2.2 *Black Box Testing***

Pengujian *black box testing* yang dilakukan dalam penelitian ini sesuai dengan tahap metode penelitian SDLC yang di rancang pada BAB III. 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 dapat di lihat pada lampiran 3. Berdasarkan hasil pengujian *black box* pada lampiran 3, dapat disimpulkan bahwa sistem aplikasi OAM dapat berjalan sesuai kebutuhan fungsional yang telah dirancang sebelumnya.

## 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. Perancangan aplikasi OAM secara konseptual dan realisasinya yang bertujuan untuk membantu masyarakat (*user*) Kota DKI Jakarta sebagai pengguna aplikasi untuk dapat melakukan pemesanan ambulans telah berhasil dilakukan. 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. Aplikasi OAM yang berjalan dengan baik sehingga dapat digunakan untuk membantu masyarakat (*user*) Kota DKI Jakarta sebagai pengguna aplikasi untuk dapat melakukan pemesanan ambulans telah berhasil dibangun. 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 terdapat 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 terdapat di DKI Jakarta. Memperbanyak titik – titik rute (*nodes*) yang terdapat didalam peta sehingga rute lebih terincikan dan memperbanyak posibilitas rute tentu dapat membantu aplikasi menjadi lebih baik dalam eksekusinya.
2. Aplikasi masih memiliki batas toleransi pengambilan titik *longitude* dan *latitude* dari GPS kurang lebih 100 m dari titik yang sebenarnya pada saat dilakukan ujicoba. Sebaiknya hal ini diperhatikan dan ditingkatkan akurasinya untuk meningkatkan kualitas aplikasi.
3. 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.
4. Menggabungkan sistem informasi (*bridging*) aplikasi OAM dan nomor KTP dengan pemerintah merupakan salah satu ide fitur aplikasi yang bagus untuk diterapkan guna mempersingkat waktu pemesanan. Sehingga dalam penerapannya, masyarakat (*user*) dapat dengan mudah mengisi seluruh informasi dasar pemesan dengan memasukkan nomor KTP nya.

## DAFTAR PUSTAKA

Ardiani, F. (2011). *Penentuan Jarak Terpendek dan Waktu Tempuh menggunakan Algoritma Dijkstra dengan Pemrograman berbasis Objek*. Yogyakarta.

AGD Dinkes. (2013). *Tentang Kami*. Dipetik Februari 2016, dari AGD Dinkes :

<http://agddinkes.jakarta.go.id/about>

Boscoe, F. P. (2007). *Geocoding Health Data, chapter 5. The Science and Art of Geocoding-Tips for Improving Match Rates and Handling Unmatched Cases in Analysis, (pp. 95–109)*. CRC Press, Amerika.

Bernstein, A. & Mutrux, Z. (2004). *Introduction to Open Source Software: How it Works, Why it's Free, and How it Might Fit the Needs of Nonprofits*.

Chamero, J. (2006). *Dijkstra's Algorithm as a Dynamic Programming strategy*.

Cormen, T. H., Leiserson, C. E., & Rivest, R. L. (1990). *Introduction to Algorithms*. MIT press, Amerika.

Connolly, T. & Begg, C. (2010). *Database Systems: a practical approach to design, implementation, and management. 5th Edition*. America: Pearson for Education.

Dewantara, Farly (2013). Perancangan Aplikasi *On-Call Positioning* Menggunakan GIS (*Geographic Information System*) dan GPS pada Dinas Pemadam Kebakaran Bandung. Skripsi, Universitas Telkom, Bandung.

Dictionary (2002). “*Ambulance*”, in *The American Heritage Science Dictionary*. Source location L Houghton Mifflin Company.

DSilva, J. (2007). *India wakes up to need for ambulance*. Dipetik Januari 2016, dari :  
<http://www.livemint.com/Politics/gtepzwzAHkk3CAVHzylUJ/India-wakes-up-to-need-for-ambulance.html>

- up-to-the-need-for-ambulances.html
- Emarketer. (2014). *2 Billion Consumers Worldwide to Get Smart(phones) by 2016.* Dipetik Januari 2016, dari Emaketer : <http://www.emarketer.com/Article/2-Billion-Consumers-Worldwide-Smartphones-by-2016/1011694>
- Fauzi, I. (2011). *Penggunaan Algoritma Dijkstra dalam pencarian rute tercepat dan rute terpendek.* Skripsi, Universitas Syarif Hidayatullah, Jakarta.
- Google Code. (t.th) *Android Anatomy and Physiology.* Google Code. Dipetik dari Januari 2016, dari : <https://sites.google.com/site/io/anatomy--physiology-of-an-android>
- Goldberg, D. W. (2011). *Improving geocoding match rates with spatially-varying block metrics.* *Transactions in GIS,* 15(6), 829–850.
- Gunandi, Y. K. & Jeffrey, T. (2002) *Perencanaan Rute Perjalanan di Jawa Timur dengan Dukungan GIS Menggunakan Metode Dijkstra's.* Jurnal Informatika Petra
- Harmon, J. E. & Anderson, S.J. (2003). *The Design and Implementation of Geographic Information Systems.* John Wiley & Sons, Inc. Canada.
- Irawan, M. P. (2011). *Perbandingan algoritma dijkstra dan algoritma bellman-ford pada jaringan grid.* Skripsi, Universitas Andalas, Padang.
- Khan, P.M., & Sufyan Beg, M.M.S. (2013). *Extended Decisions Support Matrix for Selection of SDLC-Models on Traditional and Agile Software Development Projects.* Institute of Electrical and Electronics Engineers (IEEE) Xplore Digital Library.
- Lo, C. P. and A. K. W. Yeung. (2007). *Concepts and Techniques of Geographic Information Systems.* Pearson Education Inc. 532 pp.
- Longley, Paul. A., Goodchild, Michael F., Maguire, David J., & Rhind, David W.

- (2011). *Geographic Information Systems & Science 3<sup>rd</sup> Edition*. New York, America.
- Lubis, H. S. (2009). *Perbandingan Algoritma Greedy dan Dijkstra untuk Menentukan Lintasan Terpendek*. USU Repository. Skripsi, Universitas Sumatera Utara, Medan.
- Majewski, B. (2006). *Geocoding at last!*. Dipetik April 2015, dari Maps API Blog: <http://googlemapsapi.blogspot.com/2006/06/geocoding-at-last.html>
- Maulana, A. (2015). *Jumlah pengguna Internet Indonesia Capai 88,1 Juta*. Liputan6 Tekno. Dipetik Februari 2016, dari : <http://tekno.liputan6.com/read/2197413/jumlah-pengguna-internet-indonesia-capai-881-juta>
- Munir, R. (2005). *Matematika Diskrit*. Informatika Bandung, Bandung.
- Mustaqbal, M. Sidi., Firdaus, R. Fajri., & Rahmadi, Hendra. (2015). Pengujian Aplikasi Menggunakan *Black Box Testing Boundary Value Analysis* (Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN). *Jurnal Ilmiah Teknologi Informasi Terapan Volume 1 nomor 3*. Universitas Widyatama, Bandung.
- NHS England. (2014). *Ambulance Quality Indicators Data 2014 - 15*.
- Nidhra, Srinivas. & Dondeti, Jagruthi. (2012). *Blackbox and Whitebox Testing Techniques. A literature review : International Journal of Embedded Systems and Applications (IJESA) Vol.2, No.2*.
- Novandi, R. A. D. (2007). *Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall dalam Penentuan Lintasan Terpendek (Single Pair Shortest Path)*. STEI ITB.
- Nugroho, Wahyu. (2016). *Call Operator Agd 118*.
- Oktavia, D. (t.th). *Informasi Geografis dan Informasi Keruangan*. Universitas Gunadarma.

- Panigrahy, Nilanchala. (2015). *Xamarin Mobile Application Development for Android, Second Edition*. Packt Publishing, Mumbai.
- Purwananto, Y., Purwitasari, D., & Wibowo, A. W.. (2005). *Implementasi dan Analisis Algoritma Pencarian Rute Terpendek di Kota Surabaya*. Dalam Penelitian dan Pengembangan Telekomunikasi Vol 10 No. 2:94-101. Institut Teknologi Sepuluh Noverember, Surabaya.
- Prahasta, E. (2014). *Sistem Informasi Geografis Konsep-Konsep Dasar (Perspektif Geodesi & Geomatika) Edisi Revisi*. Bandung : Informatika Bandung.
- Pressman, Roger S. (2002). *Rekayasa Perangkat Lunak Pendekatan Praktisi (Buku Satu)*. ANDI Yogyakarta.
- Reverso Dictionary. (2000). *Collin Dictionary : False*. Reverso Dictionary.
- Riyanto. (2010). *Sistem Informasi Geografis berbasis Mobile*. Gava Media, Yogyakarta.
- Robert Half. (2014). *6 Basic SDLC Methodologies: Which One is Best?*. Dipetik Januari 2017, dari :<https://www.roberthalf.com/technology/blog/6-basic-sdlc-methodologies-the-pros-and-cons>
- Safaat, N. (2012). *Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android*. Penerbit Informatika, Bandung.
- Saputro, Stevian S. (2013). *Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata Di Kota Manado Berbasis Mobile Web Dengan Algoritma Dijkstra*. Universitas Dian Nuswantoro
- Saragi, S. (2014). *Perbandingan Algoritma Dijkstra dan Floydwarshall untuk mencari Jalur Terpendek dengan Contoh Kasus Mencari Rumah Sakit*

*Terdekat di Kota Medan.* Dipetik April 2016, dari :  
<http://ust.ac.id:8080/jspui/bitstream/123456789/35/1/SAKTI%20SARAGI.pdf>

Statista. (2015). *Number of internet users in Indonesia from 2014 to 2019 (in millions).* Dipetik Januari 2016, dari : <https://www.statista.com/statistics/254456/number-of-internet-users-in-indonesia/>

Sturgeon, P. (2011). *Geocoding API's Compared.* Dipetik Maret 2016, dari :  
<https://philsturgeon.uk/2011/02/08/geocoding-apis-compared/>

Susanto, Azhar. (2004). Sistem Informasi Manajemen konsep dan pengembangannya. Lingga Jaya. Bandung.

Somani, A.K. (2005). *Survivability and Traffic Grooming in WDM Optical Networks.* Cambridge University Press, Cambridge, Inggris.

Sommerville, Ian., & Sawyer, Pete. (1997). *Requirements Engineering : A Good Practice Guide.* Wiley, Amerika.

Teske, D. (2014). *Geocoder Accuracy Ranking. In Process Design for Natural Scientists: An Agile Model-Driven Approach (pp. 161-174).* Berlin Heidelberg: Springer.

Turban, Efraim., Rainer, R., Kelly, Jr., & Potter, Richard E. (2003). *Introduction to Information Technology, second edition.* John Wiley & Sons, New-York.

Trahan, S., Nguyen, M., Aldred, I., & Jayaram., P. (2009). *Integrating Geocode Data from the Google Map API and SAS/Graph®.* North Carolina State University.

Triansyah, F. A. (2013). *Implementasi Algoritma Dijkstra Dalam Aplikasi Untuk Menentukan Lintasan Terpendek Jalan Darat Antar Kota Di Sumatera Bagian Selatan.* Jurnal Sistem Informasi (JSI), VOL. 5, NO. 2, Oktober 2013.  
Dipetik Februari 2016, dari : <http://ejournal.unsri.ac.id/index.php/jsi/index>

Vu, Phuong. (2003). *IDG Ventures Vietnam : Indonesia Mobile Game Industry*.

*Slide presentasi untuk IDG Ventures Vietnam.*

Xu, S., Flexner, S., & Carvalho, V. (2012). *Geocoding Billions of Addresses: Toward a spatial record linkage system with big data. GIScience in the Big Data Age in conjunction with the seventh International Conference on Geographic Information Science 2012*. Columbus, Ohio, USA.

## **Lampiran 1 - Wawancara**

Penulis (selanjutnya disebut dengan P) sempat melakukan wawancara dengan bapak Wahyu Nugroho, *operator call center* AGD 118 (selanjutnya disebut dengan WN). Wawancara dilakukan pada hari Senin, 14 Maret 2016. Lokasi wawancara berada di kantor AGD 118 pusat, Jalan Pahlawan Raya No.50, Ciputat Timur, Kota Tangerang Selatan, Banten 15412. Berikut pertanyaan dan jawaban hasil dari wawancara :

P : Apa itu AGD 118 ?

WN : AGD 118 merupakan sebuah instansi pelayanan kegawatdaruratan yang dimotori oleh paramedik dengan latar belakang pendidikan perawat (SPK & AKPER) dengan dukungan penuh dari IKABI IDI.

P : Dimana saja titik pos AGD 118?

WN : Saat ini pos ambulans tersebar lebih dari 20 titik di tempat berbeda.

P : Bagaimana cara pemesanan ambulans saat ini?

WN : Warga yang membutuhkan ambulans dapat langsung menelpon ke 118 atau ke pos jaga AGD 118 di nomor yang berbeda beda. Lalu memberitahukan kami informasi dasar mereka termasuk nama, alamat dan deskripsi penyakitnya.

P : Apa saja alternatif cara pemesanan ambulans saat ini?

Contohnya seperti aplikasi mobile dan web

WN : Saat ini belum ada

P : Apakah sudah ada aplikasi pemesanan ambulans secara online?

Baik di Indonesia maupun di luar negri

WN : Setahu saya di Indonesia belum ada, dan saat ini saya belum pernah mendengar ataupun melihat aplikasi tersebut

P : Data apa saja yang dibutuhkan untuk melakukan pemesanan ambulans?

WN : Data yang dibutuhkan seputar informasi dasar pasien seperti nama,

alamat, nomor telepon, keluhan. Untuk ambulans gratis dibutuhkan KTP DKI Jakarta.

- P : Bagaimana proses AGD 118 melakukan peluncuran ambulans setelah pemesanan diterima?
- WN : Kami segera melakukan koordinasi dengan pos ambulans terdekat
- P : Apa masalah masalah yang sering muncul dalam proses pemesanan ambulans?
- WN : Adanya beberapa telepon iseng (hoax)
- P : Seberapa sering terjadi *falsecall*?
- WN : Jumlah *falsecall* (hoax) dalam sehari lumayan banyak setiap harinya. Tapi tidak mencapai 50%.
- P : Apa tindakan untuk mengatasi *falsecall* tersebut?
- WN : Saat ini belum ada aktivitas yang dapat mengatasi hal tersebut.
- P : Pernahkah ada situasi dimana ketika tidak ada lagi mobil ambulans yang mengangkut akibat semua sedang dipakai? Apa solusinya?
- WN : Selama saya bekerja, AGD 118 belum pernah mengalami hal tersebut.
- P : Bagaimana respons time pada pemesanan ambulans AGD 118? Apakah ada datanya?
- WN : Response time yang baik di bawah 10 menit, namun karena banyak faktor yang terjadi dilapangan seperti kemacetan dll hal itu bisa berubah-rubah. Saat ini AGD 118 memiliki rata-rata response time 15 hingga 20 menit, pada kasus tertentu 10 menit. Kita tidak pernah lebih dari 20 menit.
- P : Apa saja langkah yang termasuk dalam response time?
- WN : Response time terhitung sejak saat pemesanan diterima hingga ambulans sampai ke tujuan.
- P : Apa teknologi yang digunakan AGD 118 saat ini?
- WN : Operator AGD 118 masih menggunakan telepon untuk menerima pemesanan ambulans.

- P : Bagaimana pendapat bapak apabila ada aplikasi yang dapat membantu masyarakat memesan ambulans?
- WN : Kami akan sangat terbantu adanya hal tersebut.
- P : Apa pendapat bapak mengenai aplikasi pesan ambulans berbasis android?
- WN : Itu merupakan salah satu inovasi yang dibutuhkan dilingkungan kesehatan kita dan dapat berguna dalam kehidupan sehari hari.
- P : Bila aplikasi pesan ambulans akan diwujudkan, menurut bapak, apa saja fitur yang dibutuhkan dalam aplikasi tersebut?
- WN : Masyarakat seharusnya dapat mendaftar dan memberikan informasi data diri mereka untuk proses pemesanannya. Dan kami dapat mendapatkan data tersebut secara otomatis.
- P : Menurut bapak, apa hal lain yang dapat menunjang aplikasi tersebut?
- WN : Mungkin anda dapat menambahkan fitur *live chat* dengan operator kami.

**Lampiran 2** - *Software Requirement Specification (SRS)*

**RANCANG BANGUN APLIKASI OAM (*ORDER AMBULANCE MOBILE*) MENGGUNAKAN GIS (*GEOGRAPHIC INFORMATION SYSTEM*) DAN ALGORITMA DIJKSTRA**

*Software Requirement Specification*

**Versi 1.0**

**30 Januari 2017**

**Atikah Chairunnisa**

*Software Engineer*

Dipersiapkan untuk

Kelengkapan Tugas Akhir Informatika Universitas Bakrie

Pembimbing : Yusuf Lestanto, S.T., M.Sc.

## **1. PENDAHULUAN**

Dokumen ini berisi penjelasan mengenai *System Requirement Specification* (SRS) yang didalamnya terdapat segala kebutuhan pembuatan aplikasi *Order Ambulance Mobile* (OAM) baik dalam kebutuhan fungsional maupun *non* fungsional. Dokumen ini dapat menjadi acuan dalam penggeraan aplikasi agar dapat berjalan dengan baik sesuai dengan yang diinginkan. Isi dokumen ini dibagi menjadi tiga bagian besar, yaitu Pendahuluan, Gambaran Umum, dan Spesifikasi Kebutuhan.

### **1.1 Tujuan**

Tujuan utama dari dokumen SRS adalah memberikan gambaran secara jelas dan rinci dari kebutuhan yang dibutuhkan dalam pembuatan aplikasi. Spesifikasi kebutuhan tersebut meliputi *software* dan *hardware*, gambaran pembuatan aplikasi, serta hal-hal yang dibutuhkan selama pembuatan aplikasi seperti kebutuhan fungsional dan *non* fungsional serta kebutuhan *interface*.

### **1.2 Ruang Lingkup**

Segala hal yang berada dalam dokumen ini merupakan batasan atau ruang lingkup dari kebutuhan pembuatan *software* berupa aplikasi OAM yang digunakan sebagai aplikasi yang membantu masyarakat di wilayah kota DKI Jakarta dalam melakukan pemesanan ambulans. Aplikasi ini dapat diakses oleh siapa saja yang membutuhkan dan harus melalui registrasi dan melakukan *login*.

### **1.3 Glossarium**

- SRS : *Software Requirement Specification*
- GIS : *Geographic Information System*

- *Software* : Perangkat Lunak
- *Hardware* : Perangkat Keras
- *Interface* : Tampilan Antar Muka
- *Pre-study* : Analisa studi pendahuluan yang dilakukan sebelum pembuatan aplikasi yang berjalan dengan tujuan untuk mengetahui masalah – masalah yang ada.

#### **1.4 *Overview***

Dokumen SRS merupakan acuan atau standar untuk menyelesaikan penggerjaan aplikasi agar sesuai dengan spesifikasi yang telah ditentukan sehingga akan menghasilkan aplikasi yang dapat memberikan manfaat serta sebagai solusi bagi masalah yang ada.

### **2. GAMBARAN UMUM**

#### **2.1 Perspektif Produk**

Aplikasi *Order Ambulance Mobile* (OAM) merupakan aplikasi berbasis *web* dan Android untuk membantu masyarakat sebagai *user* dalam melakukan pemesanan ambulans di Kota DKI Jakarta. Sistem ini menerapkan GIS dan Algoritma Dijkstra untuk mendapatkan pos ambulans terdekat dari pemesan dalam kalkulasinya. Hasil data pemesanan akan diterima oleh *website* admin ambulans dan akan segera diproses setelah dilakukan proses konfirmasi. Pengguna dari sistem ini adalah seluruh masyarakat di Kota DKI Jakarta.

#### **2.2 Fungsi Produk**

Fungsi utama dari sistem berdasarkan kebutuhan pengguna ialah sebagai berikut :

- Membantu masyarakat Kota DKI Jakarta sebagai *user* aplikasi dalam melakukan pemesanan ambulans dengan mudah melalui *smartphone* mereka.
- Membantu admin ambulans dalam mendapatkan data pemesanan sekaligus pos ambulans terdekat dari pemesan yang terkalkulasi secara otomatis menggunakan algoritma Dijkstra.

### **2.3 Karakteristik Pengguna**

Masyarakat khususnya Kota DKI Jakarta yang telah melakukan registrasi dan *log in* dapat menggunakan aplikasi ini. Autentikasi dibutuhkan agar tidak semua orang menggunakan aplikasi OAM sewenang-wenang, sehingga dapat merugikan pihak penyedia jasa ambulans.

### **2.4 Batasan Umum**

Batasan sistem aplikasi OAM dalam SRS ini antara lain:

1. Terlepas dari proses rekrutmen secara keseluruhan, sistem hanya mengakomodasi manajemen data lowongan kerja, pelamar, aplikasi lamaran, dan seleksi resume.
2. Informasi data yang dibutuhkan untuk pemesanan ialah nama, alamat (didapatkan secara otomatis menggunakan GPS), patokan alamat, deskripsi penyakit pasien, alamat *email*, dan *password*.
3. Implementasi aplikasi hanya mencakup di wilayah DKI Jakarta.
4. Penelitian tidak membahas tentang *tracking* pasien (*user*) maupun *driver* ambulans. Penelitian juga tidak mencakup jaringan dan keamanan (*security*) pada *device*.

### **2.4 Asumsi dan Ketergantungan**

Asumsi dan ketergantungan pada aplikasi adalah :

1. Pengguna (*user*) harus memiliki kemampuan untuk mengoperasikan *smart phone*.
2. Admin Ambulans harus memiliki kemampuan untuk mengoperasikan *web browser* dan komputer secara umum.
3. Informasi data yang dibutuhkan untuk pemesanan seperti nama, alamat (didapatkan secara otomatis menggunakan GPS), patokan alamat, deskripsi penyakit pasien tidak boleh kosong dan harus diisi dengan lengkap.
4. Tidak ada *training* khusus bagi *user* ataupun admin ambulans yang akan menggunakan aplikasi OAM, karena aplikasi tidak rumit dan tidak membutuhkan banyak operasi dalam penggunaannya.

### **3. SPESIFIKASI KEBUTUHAN**

#### **3.1 *Fungsionality***

1. Aplikasi membutuhkan proses registrasi, login dan logout Untuk dapat mengakses aplikasi, diperlukan aktivitas *registration*, *login* maupun *logout* karena aplikasi ini mengandung informasi yang personal sehingga perlu dibatasi aksesibilitas mengenai siapa saja dapat menggunakan aplikasi.
2. Aplikasi dapat melakukan pemesanan ambulans yang nantinya akan meminta *user* mengisi beberapa data tambahan seperti alamat (didapatkan secara otomatis ketika *user* menghidupkan fitur GPS pada *smartphone*), patokan alamat, deskripsi penyakit pasien.
3. Aplikasi dapat mengubah informasi akun seperti nama, alamat *email*, nomor hp, dan *password* pada halaman akun saya.

#### **3.2 *Usability***

1. Aplikasi dapat digunakan oleh *user* dengan mudah karena tampilan yang *user friendly* dan tidak membutuhkan banyak kegiatan

didalamnya.

2. Aplikasi dapat digunakan oleh semua orang yang memiliki *smartphone* yang di dalamnya telah ter-*install* aplikasi OAM.
3. *Smartphone* yang digunakan untuk dapat menjalankan aplikasi OAM yaitu jenis *smartphone* dengan OS Android.

### **3.3 Reliability**

Ketersediaan aplikasi ini ada setiap saat selama dibutuhkan sesuai dengan jam kerja penyedia jasa ambulans yaitu 24 jam setiap hari, dengan syarat aplikasi telah ter-*install* di *smartphone* dan memiliki jaringan koneksi internet.

### **3.4 Performance**

Cara yang dapat dilakukan jika aplikasi *error* adalah menutup aplikasi terlebih dahulu lalu beberapa saat kemudian membuka dan menjalankannya kembali.

### **3.5 Supportability**

Aplikasi ini dibangun agar dapat berjalan pada *mobile device*, sehingga *user interface* diusahakan sesederhana dan senyaman mungkin, mengingat ukuran layar *smarphone* terbatas.

### **3.6 Purchased Component**

Pada proses pembangunan aplikasi ini penulis membayar *hosting* dan *domain* agar aplikasi dapat berjalan dengan baik dan benar.

### **3.7 Kebutuhan Hardware**

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

### **3.8 Kebutuhan *Software***

Berikut merupakan daftar *software* yang dibutuhkan untuk menjalankan sistem :

- Android Studio 2.2.3
- Adobe Photoshop CS6
- Adobe Illustration CS6
- Adobe Experience Design CC 0.6.8.6 (Beta)
- MongoChef Core 4.5.2

- Gmap3
- Web Browser : Chrome (55.0.2883.95) & Safari 9.0.1 (11601.2.7.2)
- Twitter Bootstrap 3.2.1
- LoopBack 2.4.8
- NodeJS 7.2.1
- ExpressJS 4
- CoreUI Website Template

## **Lampiran 3 - Elisitasi Kebutuhan (*Requirement Elicitation*)**

Elisitasi kebutuhan (*Requirement elicitation*) merupakan salah satu tahap yang sulit dalam spesifikasi perangkat lunak disebabkan tiga masalah : masalah cakupan, masalah pemahaman, dan masalah perubahan (Nuiseibeh, 2000). Hal tersebut muncul karena pihak dengan kepentingan sering kali tidak mengetahui apa yang diinginkan dan mengungkapkan keinginannya dalam kalimat yang umum, dan beberapa pihak memiliki permintaan yang berbeda-beda yang dinyatakan dalam cara yang berbeda pula. Selain itu, faktor politik dapat mempengaruhi kebutuhan sistem. Serta, lingkungan bisnis dan ekonomi yang bersifat dinamis juga mempengaruhi tingkat kesulitan elisitasi kebutuhan (Sommerville, 2007).

Dalam mengumpulkan informasi elisitasi kebutuhan (*Requirement elicitation*) ada beberapa teknik yang dapat dilakukan, diantaranya adalah tradisional, berkelompok, dan *model driven*. Pada teknik elisitasi tradisional terdapat teknik wawancara, kuisioner, observasi, dan pengamatan dokumen. Elitisasi berkelompok terdapat teknik *brainstorming*, *Joint Application Design* (JAD), dan *prototyping*. Selanjutnya pada teknik elisitasi *model driven* terdapat *goal based* dan *scenario based*. Teknik yang dilakukan untuk pengambilan data informasi elisitasi kebutuhan pada aplikasi OAM akan menggunakan teknik elisitasi tradisional yaitu wawancara dan observasi. Elitisasi kebutuhan yang didapat melalui metode observasi dan wawancara dilakukan melalui tiga tahap yaitu :

a. Tahap I

Pada tahap ini terdapat rancangan sistem baru yang didapatkan dari hasil wawancara dengan pihak AGD 118 terkait.

b. Tahap II

Selanjutnya pada tahap ini, hasil dari elisitasi sebelumnya akan diklasifikasi menggunakan metode MDI (*Mandatory Desireable*

*Inessential*). M (*Mandatory*) berarti kebutuhan merupakan kebutuhan penting dan tidak boleh dihilangkan. D (*Desireable*) berarti kebutuhan tidak terlalu penting dan boleh dihilangkan. I (*Inessensial*) berarti kebutuhan tidak memiliki kepentingan yang berarti dengan sistem yang akan dibangun. Metode MDI ini memisahkan rancangan sistem yang dibutuhkan dengan yang disanggupi oleh penulis.

c. Tahap III

Pada tahap akhir ini, seluruh *requirement* (kebutuhan) akan diklasifikasikan menggunakan metode TOE. Metode ini terdiri atas singkatan dari Teknikal, Operasional, dan Ekonomi. Nantinya, metode ini akan dibagi kembali menjadi beberapa kelompok kesulitan seperti H (*high*), M (*middle*), dan L (*Low*).

Berikut adalah hasil elisitasi kebutuhan yang telah dilakukan :

**Tabel 3.1 Elitisasi kebutuhan (*Requirement elicitation*) tahapi I dan II untuk aplikasi OAM.**

Analisis Kebutuhan		M	D	I
1.	Menampilkan halaman registrasi aplikasi	V		
2.	Menampilkan halaman <i>log in</i> aplikasi	V		
3.	Menampilkan halaman <i>edit</i> akun	V		
4.	Menampilkan halaman pemesanan	V		
5.	Menampilkan FAQ aplikasi		V	
6.	Menampilkan <i>live chat</i> pada aplikasi		V	
7.	Menampilkan halaman proses pengambilan titik terdekat	V		
8.	Menampilkan halaman <i>log out</i> aplikasi	V		
9.	Menampilkan halaman <i>log in</i> pada website	V		
10.	Menampilkan halaman daftar pos pos ambulans dan informasinya	V		

11.	Menampilkan halaman informasi pemesanan berdasarkan status pemesanan maupun keseluruhan	V		
12.	Menampilkan halaman <i>log out</i>	V		
13.	Sistem terintegrasi dengan baik	V		

**Tabel 3.1** Elitisasi kebutuhan (*Requirement elicitation*) tahap III untuk aplikasi OAM.

Feasibility		T			O			E			
Risk		L	M	H	L	M	H	L	M	H	
Analisis Kebutuhan											
1	Menampilkan halaman registrasi aplikasi		V								
2	Menampilkan halaman <i>log in</i> aplikasi		V								
3	Menampilkan halaman <i>edit</i> akun		V								
4	Menampilkan halaman pemesanan		V								
5	Menampilkan FAQ aplikasi			V							
6	Menampilkan <i>live chat</i> pada aplikasi			V							
7	Menampilkan halaman proses pengambilan titik terdekat		V								
8	Menampilkan halaman <i>log out</i> aplikasi		V								
9	Menampilkan halaman <i>log in</i> pada website		V								
10	Menampilkan halaman daftar pos pos ambulans dan informasinya						V				
11	Menampilkan halaman informasi pemesanan berdasarkan status pemesanan maupun keseluruhan						V				
12	Menampilkan halaman <i>log out</i>		V								
13	Sistem terintegrasi dengan baik		V								

**Tabel 3.1 Kesimpulan elisitasi kebutuhan (*requirement elicitation*) untuk aplikasi OAM.**

<b>Analisis Kebutuhan : Fungsional</b>	
1.	Menampilkan halaman registrasi aplikasi
2.	Menampilkan halaman <i>log in</i> aplikasi
3.	Menampilkan halaman <i>edit akun</i>
4.	Menampilkan halaman pemesanan
5.	Menampilkan FAQ aplikasi
6.	Menampilkan <i>live chat</i> pada aplikasi
7.	Menampilkan halaman proses pengambilan titik terdekat
8.	Menampilkan halaman <i>log out</i> aplikasi
9.	Menampilkan halaman <i>log in</i> pada website
10.	Menampilkan halaman daftar pos pos ambulans dan informasinya
11.	Menampilkan halaman informasi pemesanan berdasarkan status pemesanan maupun keseluruhan
12.	Menampilkan halaman <i>log out</i>
13.	Sistem terintegrasi dengan baik
<b>Analisis Kebutuhan : Non - Fungsional</b>	
1.	<i>User Interface</i> yang simple dan <i>user friendly</i>
2.	Menggunakan bahasa baku dan istilah yang mudah dipahami

#### Lampiran 4- Deskripsi *Use Case Diagram* Aplikasi OAM

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***

<b>Use Case Name</b>	Register	
<b>Use Case ID</b>	1	
<b>Actor</b>	Masyarakat ( <i>user</i> )	
<b>Description</b>	<i>Use Case</i> ini merupakan pendaftaran akun yang dilakukan oleh masyarakat ( <i>user</i> ) untuk mendapat akses pemesanan ambulans melalui aplikasi.	
<b>Pre-Condition</b>	Masyarakat ( <i>user</i> ) membuka aplikasi OAM.	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar masyarakat ( <i>user</i> ) melakukan pendaftaran terlebih dahulu pada saat aplikasi OAM dibuka sebelum melakukan pemesanan ambulans.	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	Membuka Aplikasi OAM.	Menampilkan halaman pendaftaran ( <i>register</i> ) dan halaman <i>log in</i> .
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	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***

<b>Use Case Name</b>	Log In	
<b>Use Case ID</b>	2	
<b>Actor</b>	Masyarakat ( <i>user</i> )	
<b>Description</b>	Masyarakat ( <i>user</i> ) dapat masuk akun aplikasi <i>Order Ambulance Mobile</i> (OAM) dari <i>use case</i> ini	

<b>Pre-Condition</b>	Masyarakat ( <i>user</i> ) membuka aplikasi OAM.	
<b>Trigger</b>	<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.	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	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> .
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	Aplikasi menampilkan halaman menu utama.	

**Tabel 3.3 Deskripsi Use Case Log In**

<b>Use Case Name</b>	Log In	
<b>Use Case ID</b>	3	
<b>Actor</b>	Admin Ambulans	
<b>Description</b>	Admin Ambulans dapat masuk <i>Order Ambulance Mobile</i> (OAM) website dari <i>use case</i> ini	
<b>Pre-Condition</b>	Admin Ambulans membuka website OAM.	
<b>Trigger</b>	<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.	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>

	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> .
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	Aplikasi menampilkan halaman menu utama.	

**Tabel 3.4 Deskripsi *Use Case Log Out***

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

**Tabel 3.5 Deskripsi Use Case Log Out**

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

**Tabel 3.6 Deskripsi Use Case Melihat Informasi Akun**

<b>Use Case Name</b>	Melihat Informasi Akun	
<b>Use Case ID</b>	6	
<b>Actor</b>	Masyarakat ( <i>user</i> )	
<b>Description</b>	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	
<b>Pre-Condition</b>	Masyarakat ( <i>user</i> ) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar masyarakat ( <i>user</i> ) dapat mengetahui informasi akun aplikasi OAM.	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>

	Klik “Lihat Info Akun”	Menampilkan informasi akun berupa nama, <i>email</i> dan nomor hp.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>	Aplikasi menampilkan halaman info akun	

**Tabel 3.7 Deskripsi *Use Case* Memperbarui  
Informasi Akun**

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

**Tabel 3.8 Deskripsi *Use Case* Pesan Ambulans**

<b><i>Use Case Name</i></b>	Pesan Ambulans
<b><i>Use Case ID</i></b>	8
<b><i>Actor</i></b>	Masyarakat ( <i>user</i> )

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

**Tabel 3.9 Deskripsi *Use Case* Melanjutkan Pemesanan**

<b>Use Case Name</b>	Melanjutkan Pemesanan
<b>Use Case ID</b>	9
<b>Actor</b>	Masyarakat ( <i>user</i> )
<b>Description</b>	<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.
<b>Pre-Condition</b>	Masyarakat ( <i>user</i> ) membuka aplikasi OAM dan telah melakukan <i>log in</i> .
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar masyarakat ( <i>user</i> ) dapat melakukan pemesanan ambulans

<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	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.
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	Aplikasi menampilkan halaman pemesanan ambulans	

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

<b>Use Case Name</b>	Menentukan Pos Terdekat Menggunakan Algoritma Dijkstra	
<b>Use Case ID</b>	10	
<b>Actor</b>	Masyarakat ( <i>user</i> )	
<b>Description</b>	<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.	
<b>Pre-Condition</b>	Masyarakat ( <i>user</i> ) membuka aplikasi OAM dan telah melakukan <i>log in</i> .	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar masyarakat ( <i>user</i> ) dapat melakukan pemesanan ambulans	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	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**

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

**Tabel 3.12 Deskripsi *Use Case* Menerima Data Pemesanan Ambulans**

<b><i>Use Case Name</i></b>	Menerima Data Pemesanan Ambulans
<b><i>Use Case ID</i></b>	12

<b>Actor</b>	Admin Ambulans	
<b>Description</b>	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.	
<b>Pre-Condition</b>	Admin ambulans membuka <i>website</i> OAM dan telah melakukan <i>log in</i> .	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar admin ambulans mendapatkan informasi pemesanan ambulans	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	-	<i>Web service</i> 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.
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	<i>Web service</i> OAM mendapatkan informasi pemesanan ambulans	

**Tabel 3.13 Deskripsi *Use Case* Menerima Pemesanan Pos**

**Terdekat**

<b>Use Case Name</b>	Menerima Status Pemesanan Pos Terdekat
<b>Use Case ID</b>	13
<b>Actor</b>	Admin Ambulans
<b>Description</b>	Admin ambulans menerima data pos pemesanan ambulans terdekat, dalam radius 3 km, dari titik pemesan yang dikirimkan oleh masyarakat ( <i>user</i> ) melalui <i>smartphone</i> mereka pada <i>use case</i> ini.

<b>Pre-Condition</b>	Admin ambulans membuka <i>website</i> OAM dan telah melakukan <i>log in</i> .	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar admin ambulans mendapatkan informasi pemesanan ambulans	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	-	<i>Web service</i> OAM akan secara otomatis mendapatkan informasi pemesanan ambulans berupa status <i>handled</i> atau <i>out of coverage</i> dan pos ambulans terdekat.
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	<i>Web service</i> OAM mendapatkan informasi pemesanan ambulans	

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

<b>Use Case Name</b>	Melihat Informasi Titik Rute / Nodes dan Pos Ambulans di Jakarta	
<b>Use Case ID</b>	14	
<b>Actor</b>	Admin Ambulans	
<b>Description</b>	Pada aplikasi <i>website</i> <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.	
<b>Pre-Condition</b>	Admin ambulans membuka <i>website</i> OAM dan telah melakukan <i>log in</i> .	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar admin ambulans mendapatkan informasi titik-titik rute ( <i>nodes</i> ) dan pos ambulans di Jakarta	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>

	-	Informasi titik – titik rute /nodes dan pos – pos ambulans dapat dilihat secara langsung pada halaman menu utama website OAM.
<i>Alternate Course</i>	-	
<i>Post - Condition</i>		<i>Web service</i> OAM mendapatkan informasi titik rute nodes dan pos ambulans

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

<b><i>Use Case Name</i></b>	Mengubah ( <i>edit</i> ) Status Pemesanan Ambulans	
<b><i>Use Case ID</i></b>	15	
<b><i>Actor</i></b>	Admin Ambulans	
<b><i>Description</i></b>	Pada aplikasi <i>website 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.	
<b><i>Pre-Condition</i></b>	Admin ambulans membuka <i>website</i> OAM dan telah melakukan <i>log in</i> .	
<b><i>Trigger</i></b>	<i>Use case</i> ini dilakukan agar admin ambulans dapat mengubah status pemesanan ambulans yang tidak dapat dikonfirmasi kebenarannya menjadi <i>declined</i> .	
<b><i>Typical Of Events</i></b>	<b><i>Actor Action</i></b>	<b><i>System Response</i></b>
	-	Status pemesanan ambulans yang tidak dapat dikonfirmasi kebenarannya via telefon dapat diubah ( <i>edit</i> ) melalui <i>use case</i> ini.
<b><i>Alternate Course</i></b>	-	

<b>Post - Condition</b>	Website OAM memperbarui status pemesanan ambulans
-------------------------	---

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

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

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

<b>Use Case Name</b>	Melihat Informasi Pemesanan Ambulans dan Status Pemesanan	
<b>Use Case ID</b>	17	
<b>Actor</b>	Admin Ambulans	
<b>Description</b>	Pada aplikasi website <i>Order Ambulance Mobile</i> (OAM), admin ambulans dapat melihat informasi pemesanan ambulans dan status pemesanannya.	
<b>Pre-Condition</b>	Admin ambulans membuka website OAM dan telah melakukan <i>log in</i> .	
<b>Trigger</b>	<i>Use case</i> ini dilakukan agar admin ambulans dapat mengetahui informasi masyarakat ( <i>user</i> ) yang pernah melakukan pemesanan ambulans.	
<b>Typical Of Events</b>	<b>Actor Action</b>	<b>System Response</b>
	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.
<b>Alternate Course</b>	-	
<b>Post - Condition</b>	Website OAM mengetahui informasi masyarakat ( <i>user</i> ) yang pernah melakukan pemesanan ambulans	

## Lampiran 5

## Tabel Hasil Pengujian Aplikasi OAM

### Menggunakan Metode *White Box Testing*

Pengujian menggunakan metode *white box testing* diterapkan pada beberapa fungsi Algoritma Dijkstra. Hasil dari pengujian *white box* terhadap sebagian fungsi penting Algoritma Dijkstra pada aplikasi OAM dapat dilihat di gambar dan tabel 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 4.25 Fungsi initiateNode()

Tabel 4.1 Fungsi initiateNode()

Nama Fungsi	initiateNode()
Deskripsi	Fungsi ini digunakan untuk mengetahui seluruh titik – titik hasil respon dari <i>web service</i> . Lihat gambar 4.25 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/ <i>node</i> dalam variabel <i>arrayListNode</i> seperti yang terlihat pada <i>source code</i> ( <i>jsonObjectNodePosition.put(strNama, i)</i> ).
Status	Succeed

```

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 4.26 Fungsi getAllVertex()**

**Tabel 4.2 Fungsi getAllVertex()**

<b>Nama Fungsi</b>	getAllVertex()
<b>Deskripsi</b>	Fungsi ini digunakan untuk mengetahui titik tetangga dari seluruh titik yang diketahui dari hasil respon dari <i>web service</i> . Lihat gambar 4.26 untuk fungsi getAllVertex()
<b>Input</b>	<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.
<b>Output</b>	<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.
<b>Status</b>	<b>Succeed</b>

```

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 4.27 Fungsi sendRequest()**

**Tabel 4.3 Fungsi sendRequest()**

<b>Nama Fungsi</b>	sendRequest()
<b>Deskripsi</b>	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> ).
<b>Input</b>	<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 4.27 yang menunjukkan fungsi sendRequest()
<b>Output</b>	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 <i>source code</i> pada gambar Gambar 4.27 yang menunjukkan fungsi <i>getMarkerFlat(boolean is_pos, int intPosition)</i> .
<b>Status</b>	Succeed

```

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", "testggs" + 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 4.28 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 4.29 Fungsi execute() dan traceBack()**

**Tabel 4.4 Fungsi execute()**

<b>Nama Fungsi</b>	execute ()
<b>Deskripsi</b>	Fungsi ini digunakan untuk menentukan titik aktif pada peta dan melihat tetangganya. Kemudian mengambil nilai terbaik

	<p>/ 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 4.29 untuk fungsi execute() dan fungsi traceBack()</p>
<b>Input</b>	Titik awal node (strAwal). Titik strAkhir berasal dari arrayListPos (array yang bervariabel string) yang dipecah menjadi masing masing strAkhir. Sehingga apabila didalam arrayListPos terdapat 3 Pos maka proses dijkstra akan dilakukan sebanyak 3 kali ke masing target pos tersebut. jsonObjectNodePosition, salah satu input untuk fungsi execute(), merupakan json object yang berisikan key value informasi posisi index node pada arrayListNode, arrayListNode, merupakan array yang berisikan variabel Node.
<b>Output</b>	JsonObjectResult, merupakan json object yang berisikan key path dan value jsonArrayPath. jsonArrayPath merupakan json array dari variabel 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.
<b>Status</b>	Succeed

**Lampiran 6****Tabel Hasil Pengujian Aplikasi OAM****Menggunakan Metode *Black Box Testing***

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 dapat di lihat pada tabel 4.5 dan tabel 4.6.

**Tabel 4.5 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 pertama yang menampilkan <i>log in</i> dan <i>register</i>	Muncul halaman pertama yang menampilkan <i>log in</i> dan <i>register</i>	Berhasil
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 klik tombol “lanjutkan pesanan”	Aplikasi menampilkan halaman pencarian rute terdekat menggunakan Algoritma Dijkstra.	Aplikasi menampilkan halaman pencarian rute terdekat menggunakan Algoritma Dijkstra.	Berhasil
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	Klik <i>textbox</i> nomor hp dan ubah	Aplikasi akan menampilkan langsung data	Aplikasi akan menampilkan langsung data	Berhasil

	pada halaman Akun Saya	nomor hp yang diinginkan >Klik “Ubah data”.	yang telah diubah.	yang telah diubah.	
9	Melakukan <i>edit password</i> pada halaman Akun Saya	Klik <i>textbox 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 alamat email</i> pada halaman Akun Saya	Klik <i>textbox alamat email</i> dan ubah alamat <i>email</i> yang diinginkan >Klik “Ubah data”.	Aplikasi akan menampilkan langsung data yang telah diubah.	Aplikasi akan menampilkan langsung data yang telah diubah.	Berhasil
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 4.6 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	Muncul halaman pertama yang	Berhasil

			menampilkan <i>log in</i>	menampilkan <i>log in</i>	
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 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..	<i>Website</i> menampilkan halaman <i>main menu</i> . Dalam <i>main menu</i> terdapat informasi jumlah order, jumlah pengguna 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..	Berhasil
3	Melihat order pemesanan ambulans	Klik order > All	<i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	<i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	Berhasil
4	Melihat order pemesanan ambulans	Klik order > <i>Handled</i>	<i>Website</i> akan menampilkan informasi pemesanan dengan status <i>handled</i> berupa tanggal,	<i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude	Berhasil

			waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	dan latitude, status pemesanan, dan pos terdekatnya.	
5	Melihat order pemesanan ambulans	Klik order > <i>Unhandled</i>	<i>Website</i> akan menampilkan informasi pemesanan dengan status <i>unhandled</i> berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	<i>Website</i> akan menampilkan seluruh informasi pemesanan berupa tanggal, waktu, alamat, titik longitude dan latitude, status pemesanan, dan pos terdekatnya.	Berhasil
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 longitude dan latitude, serta nomor teleponnya.	<i>Website</i> akan menampilkan informasi pos – pos ambulans di Jakarta berupa nama pos, alamat, titik longitude dan latitude, serta nomor teleponnya.	Berhasil
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 <i>website</i> 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