

PROPOSAL TUGAS AKHIR

**RANCANG BANGUN APLIKASI OAM (*ORDER AMBULANCE ONLINE*)
MENGUNAKAN METODE GIS (*GEOGRAPHIC INFORMATION SYSTEM*)
DAN ALGORITMA DIJSKTRA**



ATIKAH CHAIRUNNISA

1112001018

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS BAKRIE
JAKARTA
2016**

HALAMAN PENGESAHAN

1. Judul Penelitian : Rancang Bangun Aplikasi OAM (*Order Ambulance Mobile*) Menggunakan Metode GIS (*Geographic Information System*) dan Algoritma Dijkstra
2. Bidang Penelitian : Informatika
3. Peneliti Utama : Atikah Chairunnisa
4. Jenis Kelamin : Perempuan
5. Unit Kerja : Program Studi Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Bakrie
6. Alamat/Telp. Unit kerja : Gelanggang Mahasiswa Soemantri Brojonegoro Suite GF-22. Jl. H.R. Rasuna Said Kav. C-22, Jakarta Selatan, Tel: 021-526 1448 Ext 243
7. Alamat/Telp.Rumah : Jl. Salemba Bluntas B7, Jakarta Pusat.
8. Alamat e-mail : msy.tika@gmail.com
9. Telepon seluler : 0811768949
10. Usulan Penelitian Tahun : 2016

Jakarta,

Menyetujui,

Dosen Pembimbing,

Peneliti,

Yusuf Lestanto, ST.,M.Sc

Atikah Chairunnisa

**RANCANG BANGUN APLIKASI OAM (*ORDER AMBULANCE MOBILE*)
MENGUNAKAN METODE 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. Salah satu kendala yang ditemui dalam proses pemesanannya di antaranya adalah *falsecall*, 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 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, GPS, Android, OAM, Pemesanan Ambulans, Algoritma Dijkstra*

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

Atikah Chairunnisa

Abstract

The location and time when the citizens in an emergency, experience life-threatening conditions, are often unpredictable. When undesirable things occurs, citizens are required to order ambulance manually by phone. One of many obstacles encountered in the process of ordering ambulances is 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 , have highee chances to be realized. This study overcome the problem of retrieval efficiency and data processing applications from patients using OAM (Order Ambulance Mobile). The applications that will be used is android based. 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, GPS, Android, OAM, Order Ambulance, Dijkstra Algorithm*

DAFTAR ISI

HALAMAN JUDUL.....	i
HALAMAN PENGESAHAN.....	ii
ABSTRAK.....	iii
ABSTRACT.....	iv
DAFTAR ISI.....	v
DAFTAR TABEL.....	vii
DAFTAR GAMBAR.....	viii
DAFTAR SINGKATAN.....	ix
BAB I. PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah.....	5
1.3 Batasan Masalah.....	5
1.4 Tujuan Penelitian.....	6
1.5 Manfaat Penelitian.....	6
1.6 Sistematika Penulisan Tugas Akhir.....	6
BAB II. LANDASAN TEORI.....	viii
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.....	19
2.3.1 Anatomi Android.....	20
2.4 Algoritma Pencarian Jarak Terpendek (<i>Shortest Path Algorithm</i>).....	24
2.4.1 Algoritma Dijkstra.....	25
2.4.2 Algoritma Bellman Ford.....	28
2.4.3 Algoritma Floyd Warshall.....	30

2.4.3	Perbandingan Algoritma Dijkstra, Bellman Ford, dan Floyd Warshall.....	31
2.5	<i>Mobile Application Development Life Cycle (MADLC)</i>	33
BAB III. METODE PENELITIAN.....		33
3.1	Perancangan Aplikasi.....	33
3.2	Jenis Penelitian.....	47
3.3	Objek Penelitian.....	48
3.4	Rencana Kegiatan Penelitian.....	48
DAFTAR PUSTAKA.....		49

DAFTAR TABEL

Tabel 2.1	Penelitian Terdahulu dan Penelitian Penulis.....	9
Tabel 2.2	Bentuk-bentuk data informasi lokasi (spasial)	13
Tabel 2.3	Perbandingan <i>online geocoding</i> API	18
Tabel 2.4	Algoritma Dijkstra dalam bentuk <i>Pseudocode</i>	27
Tabel 2.5	Algoritma Bellman Ford.....	29
Tabel 2.6	Algoritma Floyd Warshall.....	30
Tabel 2.7	Perbandingan algoritma pencari jarak terpendek (<i>shortest path algorithm</i>)	31
Tabel 3.1	Deskripsi <i>Use Case</i> Diagram Aplikasi OAM.....	35
Tabel 3.2	Deskripsi simbol – simbol <i>Class Diagram</i>	38
Tabel 3.3	Deskripsi <i>Multiplicity</i> pada <i>Class Diagram</i>	39
Tabel 3.4	Rencana Kegiatan Penelitian.....	51

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.....	21
Gambar 2.5	Klasifikasi graf berdasarkan orientasinya; (a) Graf terarah / <i>Directed graph</i> (b) Graf tidak terarah / <i>Undirected graph</i>	25
Gambar 2.6	Diagram proses pada MADLC.....	33
Gambar 3.1	<i>Use Case Diagram</i> Aplikasi OAM.....	37
Gambar 3.2	<i>Activity Diagram</i> pada Aplikasi OAM.....	39
Gambar 3.3	<i>Class Diagram</i> pada Aplikasi OAM.....	40
Gambar 3.4	<i>Entity Relationship Diagram</i> (ERD) pada Aplikasi OAM.....	41
Gambar 3.5	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Satu.....	42
Gambar 3.6	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Dua.....	42
Gambar 3.7	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Tiga.....	43
Gambar 3.8	Perancangan Antarmuka Menu Utama pada Aplikasi OAM Bagian Empat.....	43
Gambar 3.9	Perancangan Antarmuka Informasi Aplikasi pada Aplikasi OAM.....	44
Gambar 3.10	Perancangan Antarmuka Akun Saya pada Aplikasi OAM	44

DAFTAR SINGKATAN

ADT	<i>Android Development Tools</i>
AGD	<i>Ambulans Gawat Darurat</i>
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>
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 di ikuti 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 yang mendominasi populasi pengguna *smartphone* di Indonesia.

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

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

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

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

Pada saat wawancara dengan bapak Wahyu Nugroho, *call center operator* AGD 118, beliau mengatakan bahwa *response time* penanganan ambulans di AGD 118 tercatat dengan rata-rata lebih dari 10 menit. AGD 118 berbasis di Jalan Pahlawan Raya No.50, Banten. Sejak tahun 2006 hingga saat ini, AGD 118 memiliki catatan *response time* 10 hingga 20 menit. Hal ini menunjukkan bahwa *response time* merupakan catatan yang penting dan sulit ditekan. 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 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 Perencanaan Rute Perjalanan di Jawa Timur dengan Dukungan GIS menggunakan Metode Dijkstra, karena hasil yang dikeluarkan oleh algoritma tersebut hanya satu rute saja yang merupakan rute terpendek dari satu kota ke kota lain di dalam penelitian tersebut (Gunandi, 2002).

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

1.2 Rumusan Masalah

Dalam penelitian ini masalah yang dibahas adalah sebagai berikut:

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

1.3 Batasan Masalah

Batasan dari penelitian ini adalah sebagai berikut:

1. Aplikasi ini berjalan pada smartphone Samsung galaxy S5
O.S Android versi 4.4.2, Kitkat.
2. Implementasi aplikasi hanya mencakup di wilayah DKI Jakarta
3. Server menggunakan hosting dari Domainsia
4. Menggunakan GPS bawaan dari smartphone
5. Keamanan Data tidak terlalu diperhatikan Aplikasi tidak membahas tentang *tracking* pasien (*user*)
6. Jaringan pada device tidak terlalu diperhatikan
7. Penentuan ketersediaan ambulans tiap pos ambulans dilakukan secara manual

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

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

1.5 Manfaat Penelitian

Manfaat dari penelitian yang penulis lakukan adalah sebagai berikut:

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

BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

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

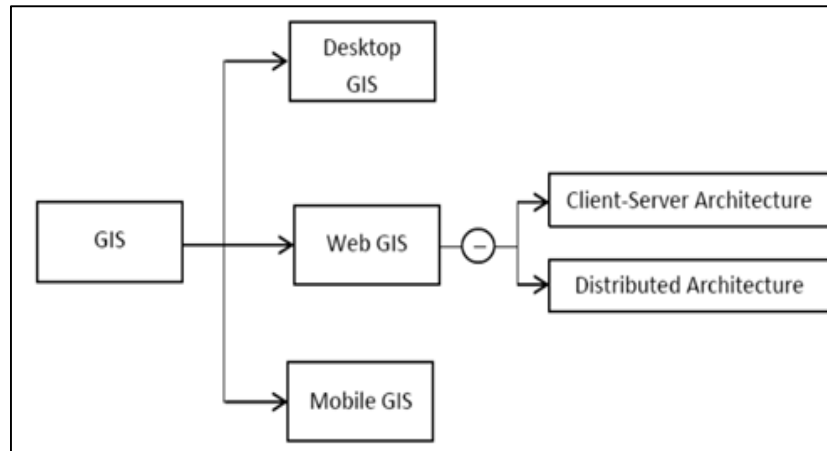
1. Penelitian yang dilakukan oleh Farly Nur Dewantara dengan judul penelitian *Perancangan Aplikasi On-Call Positioning Menggunakan GIS (Geographic Informarion System) dan GPS pada Dinas Pemadam Kebakaran Bandung*. Penelitian ini menghasilkan aplikasi mobile pelaporan kebakaran kepada petugas pemadam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

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

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

2.2 GIS (*Geographic Information System*)

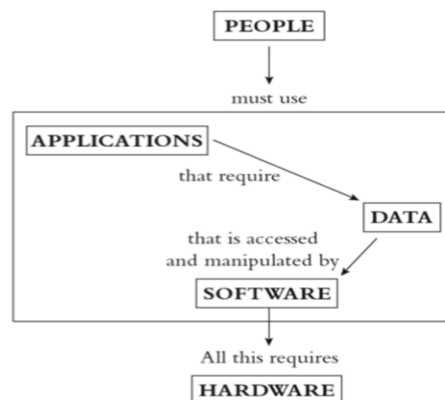
GIS atau *Geographic Information System* (dalam bahasa Indonesia disebut Sistem Informasi Geografis) merupakan sistem informasi yang mengelola data yang memiliki informasi spasial. GIS merupakan sistem komputer yang memiliki kemampuan khusus untuk membangun, menyimpan, mengelola dan menampilkan informasi be referensi geografis (Riyanto, 2010). Sistem ini berkaitan erat dengan informasi geografis sehingga ada juga yang mendefinisikan GIS sebagai sistem informasi yang dapat memberikan informasi lokasi dan waktu mengenai suatu kejadian atau aktivitas (Longley, 2011). Berdasarkan teknologi dan implementasinya, GIS (*geographic information system*) dapat dikategorikan dalam 3 aplikasi yaitu GIS berbasis *desktop* (*Desktop GIS*), GIS berbasis *web* (*Web GIS*), dan GIS berbasis *mobile* (*Mobile GIS*). Lihat gambar 2.1. Ketiganya 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).



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 informasi lokasi (spasial) dan informasi deskriptif (atribut).

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

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

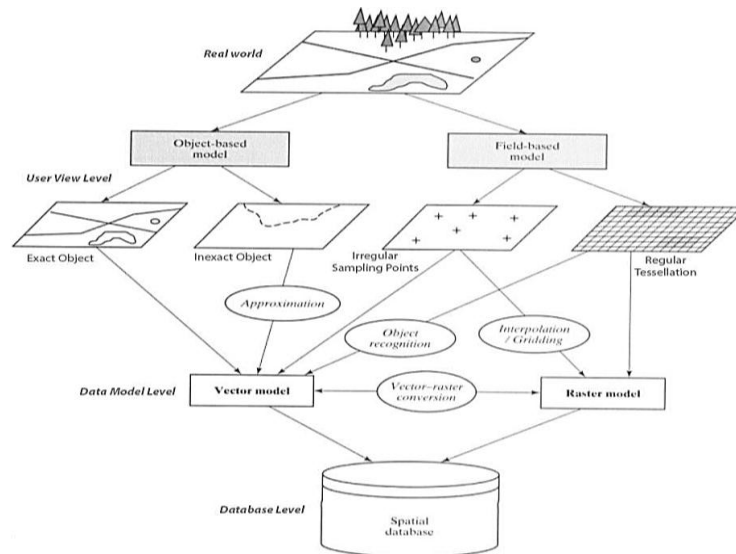
Bentuk-bentuk informasi lokasi	Deskripsi	Format Titik
Titik (<i>Dot</i>)	<p>Bentuk data ini merupakan bentuk data paling sederhana bagi obyek spasial. Tipe data ini tidak memiliki dimensi.</p> <p>Contoh :</p> <p>Lokasi kecelakaan</p>	<p>Format titik :</p> <p>Koordinat tunggal.</p> <p>Tanpa panjang/jarak.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

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

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

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

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



Gambar 2.3 Vektor dan Raster data model

(Lo, 2007)

1. Data Vektor

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

2. Data Raster

Data raster merupakan data yang merepresentasikan obyek geografis sebagai struktur sel grid yang disebut *pixel* (*picture element*). Baik digunakan untuk menggambarkan batas-batas yang berubah secara gradual, misalnya tanah, kelembaban tanah, suhu, dan sebagainya. Kelemahan data ini adalah keterbatasan data akibat besarnya ukuran *file* yang dapat mempengaruhi kualitas resolusinya. Data raster juga

bergantung pada kemampuan *hardware* dalam pengerjaannya.

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

D. Perangkat lunak (*software*)

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

E. Perangkat keras (*hardware*)

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

2.2.2 *Geocoding dan Geocoder*

Geocoding merupakan proses pengambilan data koordinat geografis berupa latitude dan 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.

Pengambilan data seperti garis bujur dan garis lintang dari nama lokasi dalam penelitian pembuatan aplikasi OAM ini akan menggunakan Google *geocoding*, karena *geocoder* ini memiliki beberapa keunggulan. Google API (*Application Programming Interface*) tidak mengharuskan melakukan pemisahan data dalam komponen tertentu, seperti jalan dan kota, karena sudah ada proses *parsing* didalamnya. Google *geocoding* juga memiliki akurasi yang bagus (Sturgeon, 2011). Para peneliti di RTI *International, North Carolina*, telah menggunakan Google API untuk melakukan banyak bentuk kalkulasi GIS (*geographic information system*) pada penelitian mereka (Trahan, 2009). Google *geocoding* juga memiliki keunggulan lain, karena di dalam *geocoder* ini juga sudah terdapat proses normalisasi dan standardisasi huruf kapital, tanda baca, dan singkatan / kependekan (Majewski, 2016).

Tabel 2.3 Perbandingan *Online Geocoding API*

(Xu, 2012)

	Google Geocode Service	Bing Geocode Service	Yahoo! Geocode Service
Contoh API request	http://maps.googleapis.com/maps/api/geocode/xml?address=500+108th+ave+ne,+bellevue+wa+98004&sensor=true	http://dev.virtualearth.net/REST/v1/Locations/US/WA/98004/bellevue/500+108th+ave+ne?key=[APIkey]	http://yboss.yahooapis.com/geo/placefinder?location=701+First+Ave,+Sunnyvale,+CA
Sumber data <i>(Data resources)</i>	Internal Street Networks	TeleAtlas, Navteq, Map Data Sciences	Navteq
Kapasitas request <i>(Request capacity)</i>	2500 <i>request</i> per hari	<i>Trial</i> - 10000 <i>request</i> per bulan <i>Basic</i> - 125000 <i>request</i> per tahun <i>Enterprise</i> - lebih dari 125000 <i>request</i> per tahun	Dapat mencapai 35000 (dan lebih dari itu) <i>request</i> per hari, tergantung pada biaya yang ditawarkan
Presisi geocoding	Rooftop, Geometric_Center,	Parcel, Interpolation, Rooftop,	99, 90, 87, 86, 85, 84, 82, 80, 75, 74, 72, 71, 70, 64,

<i>(Geocoding precision)</i>	Range_Interpolated, Approximate, Zero_Results	InterpolationOffset, Null	63, 62, 60, 59, 50, 49, 40, 39, 30, 29, 20, 19, 10, 9 , 0
<i>Return Format</i>	JSON/XML	JSON/XML	XML

2.3 Android

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

1. Lengkap (*Complete Platform*)

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

2. Terbuka (*Open Source Platform*)

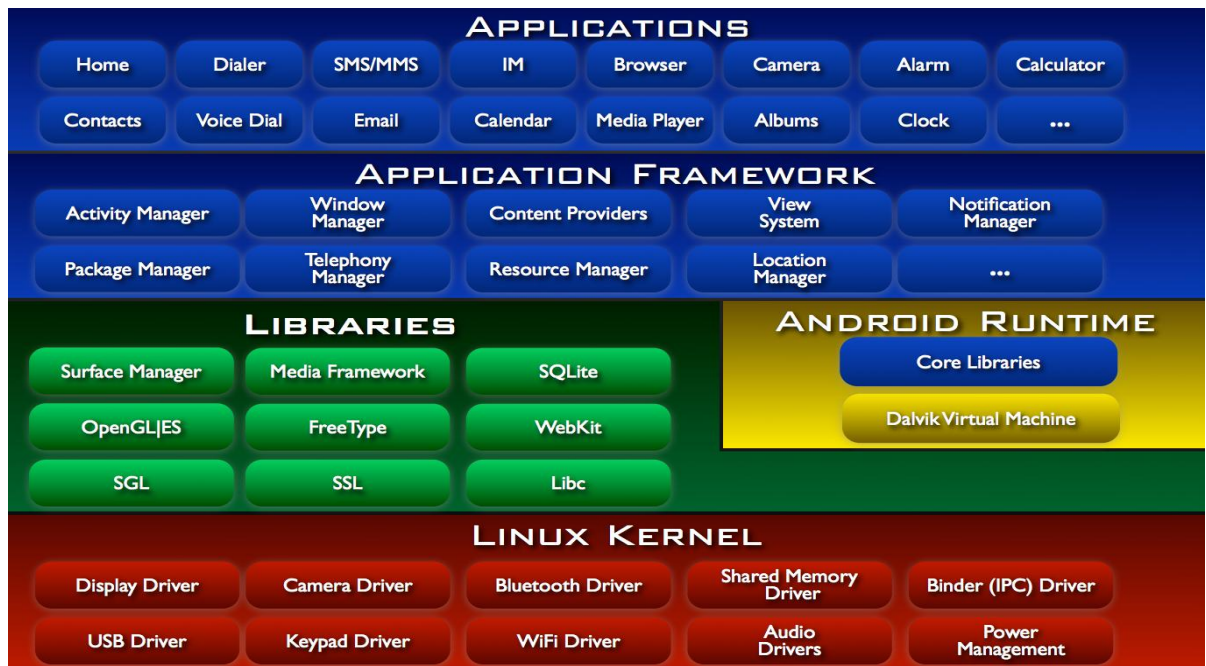
Platform android disediakan melalui lisensi *open source*. Hal itu berarti, pengembang dapat dengan bebas untuk mengembangkan aplikasi. Aplikasi dengan lisensi *open source* biasanya lebih stabil, tidak memakan biaya yang banyak / tanpa biaya, memiliki komunitas (*support community*) yang besar sehingga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

3. Free (*Free Platform*)

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

2.3.1 Anatomi Android

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



Gambar 2.4 Anatomi Android
(Google Code)

1. Linux Kernel

Meskipun android diciptakan menggunakan Linux kernel, namun android bukanlah linux. Hal itu berarti, android tidak memenuhi seluruh standar pada linux *utilities* karena android bukanlah linux. Linux kernel digunakan untuk menciptakan android karena linux kernel memiliki memori dan manajemen

proses yang baik. Linux mendukung *shared libraries* dan merupakan lisensi *opensource*. Kernel linux menyediakan driver layar, kamera, keypad, WiFi, Flash Memory, audio, dan IPC (Interprocess Communication) untuk mengatur aplikasi dan lubang keamanan .

2. Native Libraries

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

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

3. Android Runtime

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

- a) Pustaka Inti (*Core Libraries*)
Pustaka inti Android menyediakan hampir semua fungsi yang terdapat pada pustaka Java serta beberapa pustaka khusus Android.

b) Mesin Virtual Dalvik (*Dalvik Virtual Machine*)

Dalvik merupakan sebuah mesin virtual yang dikembangkan oleh Dan Bornstein yang terinspirasi dari nama sebuah perkampungan yang berada di Iceland. Dalvik hanyalah interpreter mesin virtual yang mengeksekusi file dalam format Dalvik Executable (*.dex). Dalvik *virtual machine* dapat mendukung banyak pemrosesan pada satu *device* karena menggunakan *runtime memory* secara sangat efisien.

4. *Application Framework*

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

a. *Activity Manager*

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

b. *Content Providers*

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

c. *Resource Manager*

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

d. *Location Manager*

Berfungsi untuk memberikan informasi detail mengenai lokasi perangkat Android berada.

e. *Notification Manager*

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

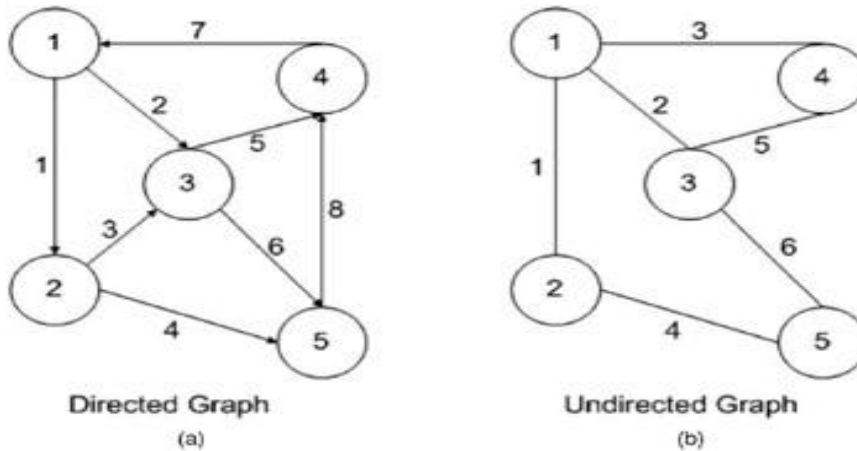
5. *Application Layer*

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

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

Algoritma pencarian jarak terpendek (*shortest path algorithm*) merupakan solusi untuk mencari jarak terpendek pada lintasan atau rute dari titik awal hingga titik tujuan. Biasanya, untuk memudahkan penggambaran ilustrasinya digunakan graf (Magzhan, 2013). Sebuah graf merupakan objek matematika yang bersifat abstrak. Graf didefinisikan sebagai pasangan himpunan (V, E) , ditulis dengan notasi $G = (V, E)$. Dalam hal ini, V (*vertices* atau *nodes*) merupakan himpunan yang tidak kosong dari simpul-simpul. Lalu E (*edge* atau *arcs*) merupakan himpunan sisi yang menghubungkan sepasang simpul. Simpul (*vertex*) dapat dinyatakan sebagai huruf, bilangan asli 1, 2, 3,...dst, atau gabungan keduanya (Munir, 2005). Sedangkan sisi yang menghubungkan simpul u dengan simpul v dinyatakan dengan (u, v) . Jika e merupakan bagian sisi yang menghubungkan pasangan (u, v) , maka $e = (u, v)$. Secara umum klasifikasi pada graf berdasarkan orientasi arah pada sisinya dibagi atas graf ber arah (*directed graph*) dan graf tidak ber arah (*undirected graph*) (Fauzi, 2011). Gambar 2.11 (a) menunjukkan gambar graf berarah karena memiliki urutan lintasan yang harus

diperhatikan. Gambari 2.11 (b) menunjukkan gambar graf yang tidak terarah, sehingga urutannya tidak diperhatikan.



Gambar 2.5 Klasifikasi graf berdasarkan orientasinya;

(a) Graf terarah / *Directed graph*

(b) Graf tidak terarah / *Undirected graph*

(Somani, 2005)

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

2.4.1 Algoritma Dijkstra

Algoritma Dijkstra ditemukan oleh Edsger Dijkstra. Algoritma ini merupakan algoritma yang paling sering digunakan dalam pencarian rute terpendek. Penggunaannya dengan menggunakan simpul simpul sederhana pada jaringan jalan yang tidak rumit menjadikannya algoritma yang sederhana untuk digunakan (Chamero,

2006). Algoritma dijkstra merupakan salah satu varian dari algoritma greedy, sehingga memiliki beberapa elemen penyusun algoritma *greedy* (Novandi, 2007) yaitu :

a. Himpunan Kandidat C

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

b. Himpunan Solusi S

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

c. Fungsi Seleksi

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

d. Fungsi Kelayakan

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

e. Fungsi Objektif

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

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

iterasi selanjutnya (Tiansyah, 2013). Salah satu cara penulisan algoritma dijkstra dapat dilihat pada tabel 2.4 sebagai berikut.

Tabel 2.4 Algoritma Dijkstra dalam bentuk *pseudocode*

(Munir, 2005 hal.414)

procedure Dijkstra (INPUT m : matriks dan a : simpul awal)
(Mencari lintasan terpendek dari simpul awal a ke semua simpul lainnya Masukan : matriks ketetanggaan (m) dari graf berbobot G dan simpul awal a Keluaran : lintasan terpendek dari a ke semua simpul lainnya)
<u>Deklarasi</u> s1, s2, ..., sn :integer (tabel integer) d1, d2, ..., dn :integer (tabel integer) i, j, k: integer
<u>Algoritma (langkah 0 / Inisialisasi)</u> for i \leftarrow 1 to n do $s_i \leftarrow 0$ $d_i \leftarrow m_{ai}$ Endfor

Algoritma (langkah 1)

Sa \leftarrow 1 (karena simpul a adalah simpul asal lintasan terpendek, jadi simpul a sudah pasti terpilih dalam lintasan terpendek)

Daoo (tidak ada lintasan terpendek dari simpul a ke a)

Algoritma (langkah 2, 3, ..., n-1:)

For k \leftarrow 2 to n-1 do

J \leftarrow simpul dengan $s_j = 0$ dan d_j minimal

Sj \leftarrow 1 {simpul j sudah terpilih ke dalam lintasan terpendek}

{perbaharui tabel d}

For semua simpul I dengan $s_i = 0$ do

If $d_j + m_{ji} < d_i$ then

Didj+mji

Endif

Endfor

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 $w : E \rightarrow \mathbb{R}$. 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 Floyd 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
(Sagih, 2014)

Algoritma Floyd : Proses perhitungan <i>shortest path</i>	
Floyd-Warshall(d,n)	
1.	for k \leftarrow 1 to n
2.	for j \leftarrow 1 to n
3.	for i \leftarrow 1 to n

4.	$d_{ij}(k) = \min(d_{ij}(k-1), d_{ij}(k-1))$
5.	if choose $d_{ij}(k-1) + d_{ij}(k-1)$ then pred(ij) = k
Algoritma Floyd : Proses penentuan rute <i>shortest path</i>	
Path (I, j)	
1.	If pred(I, j) = nil then
2.	Return output(I, j)
3.	Else
4.	Path(i, pred(I, j))
5.	Path(pred(I, j), j)

2.4.4 Perbandingan Algoritma Dijkstra, Bellman Ford, dan Floyd Warshall

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

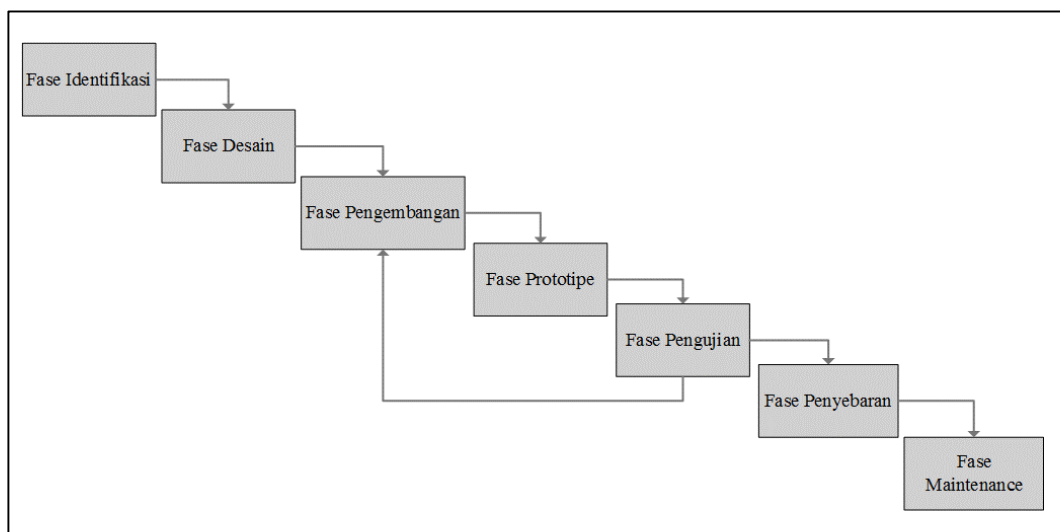
Tabel 2.7 Perbandingan algoritma pencarian jarak terpendek
(shortest path algorithm)
(Fauzi, 2012)

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

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

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

2.5 Mobile Application Development Life Cycle (MADLC)



Gambar 2.6 Diagram proses pada MADLC

(Vithani & Kumar, 2014)

Mobile Application Development Life Cycle atau MADLC merupakan metode pengembangan aplikasi yang dikhususkan untuk membangun *Mobile Apps* (Vithani & Kumar, 2014). Ada beberapa fase – fase pembangunan pada MADLC, lihat gambar diagram proses MADLC pada gambar 2.6.

1) Fase Identifikasi

Fase ini mengumpulkan ide-ide aplikasi dari yang dibutuhkan oleh *client*, kemudian *developer* dapat menganalisa kebutuhan dan menciptakan aplikasi yang baru. Jika aplikasi yang dijelaskan sebelumnya sudah ada di pasaran, 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 pasaran akan dilakukan studi komparasi kemudian dilakukan pengembangan

2) Fase Desain

Pada fase ini, *developer* memikirkan bagaimana aplikasi ini akan dibangun, *platform mobile* apa saja yang akan dipakai untuk menjalankan aplikasi. Fase 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 pengembangan aplikasi.

3) Fase Pengembangan

Fase ini terbagi menjadi dua tahap yaitu *coding* untuk fungsi aplikasi dan *coding* untuk tampilan atau UI aplikasi. Fase ini merupakan tahap pemrograman aplikasi yang merealisasikan apa yang telah direncanakan pada tahap perancangan.

4) Fase Prototype

Pengujian secara fungsional pada *coding* yang telah dilakukan pada tahap pengembangan oleh *client* dilakukan pada fase ini. *Client* akan memberikan *feedback* atas hasil dan hasil *feedback* akan dibuat *prototype* dan diberikan

kembali ke *client* hingga dapat fungsi yang diharapkan dapat berjalan sesuai dengan keinginan *client*.

5) Fase Pengujian

Fase ini akan menerapkan fungsi-fungsi aplikasi yang telah diuji pada fase *prototype* pada perangkat *mobile* yang sesungguhnya. Setelah pengujian dilakukan, hasil pengujian didokumentasi dan meminta *feedback* dari *client*.

6) Fase Penyebaran

Setelah melakukan proses pengujian dan mendapat *feedback* dari *client*, aplikasi akan masuk pada fase ini. Pada fase ini, aplikasi siap untuk disebar ke tempat yang luas, seperti *Google Play Store* untuk android atau websites dan media *social media* lainnya.

7) Fase Maintenance

Fase ini merupakan fase terakhir pada MADLC. *Feedback* dari pengguna aplikasi yang telah mengunduh / menggunakan aplikasi akan terkumpulkan pada fase ini. Fase ini juga merupakan fase untuk memperbaiki *bugs* yang ada dan *updates*.

BAB III

METODE PENELITIAN

3.1 Perancangan Aplikasi

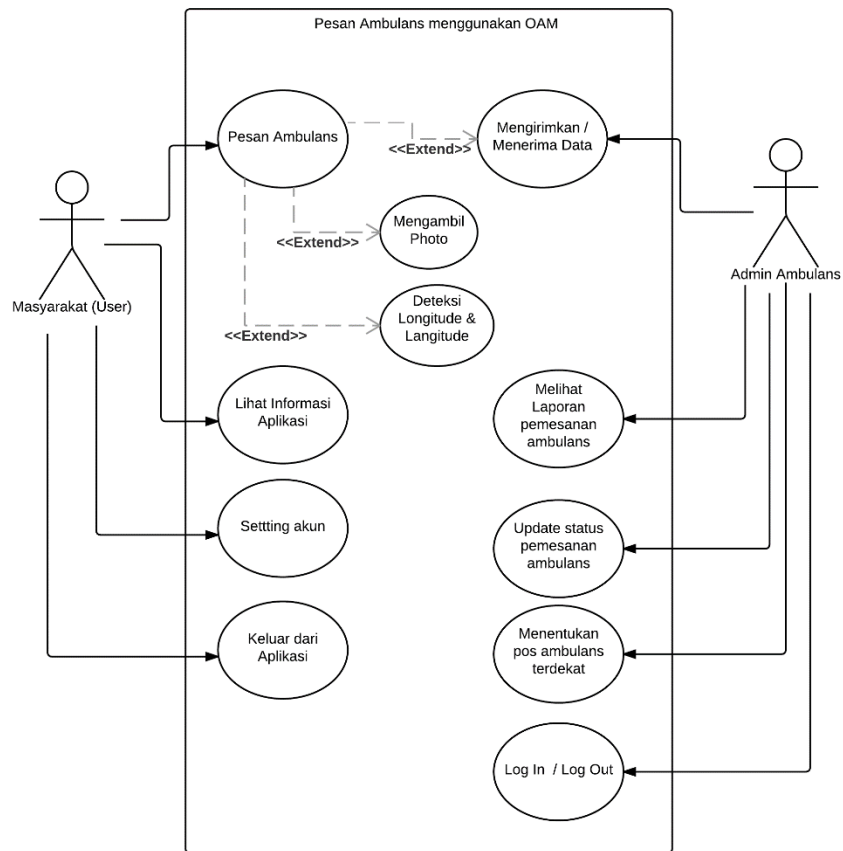
Perancangan aplikasi akan menerapkan metode MDLC yang seperti telah dijelaskan pada BAB II. MADLC merupakan metode pengembangan khusus yang untuk *mobile* yang telah diusulkan oleh Tejas Vithani dan Anand Kumar dikarenakan kompleksitas fungsi pada aplikasi *mobile*.

1) Fase Identifikasi Aplikasi

Fase ini merupakan fase tahap pertama untuk menentukan ide penelitian. Fase ini di sertai dengan observasi dari para peneliti – peneliti terdahulu, menganalisa dari berbagai sumber yang *legitimate* secara *online* maupun *offline*, seperti membaca paper, journal dan sumber lainnya untuk mengembangkam aplikasi yang sudah pernah dibuat oleh pembuat sebelumnya. OAM di buat berbeda dengan beberapa peneliti sebelumnya yaitu OAM bergerak di bidang kesehatan, menggunakan bahasa pemrograman android, dan menggunakan algoritma Dijkstra. Fase selanjutnya adalah fase desain aplikasi.

2) Fase Desain Aplikasi

Fase ini merupakan hasil observasi dari beberapa sumber untuk pengumpulan ide dan wawasan pada fase identifikasi. Pada fase ini dibuat desain 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.

a. *Use case diagram***Gambar 3.1 Use Case Diagram Aplikasi OAM**

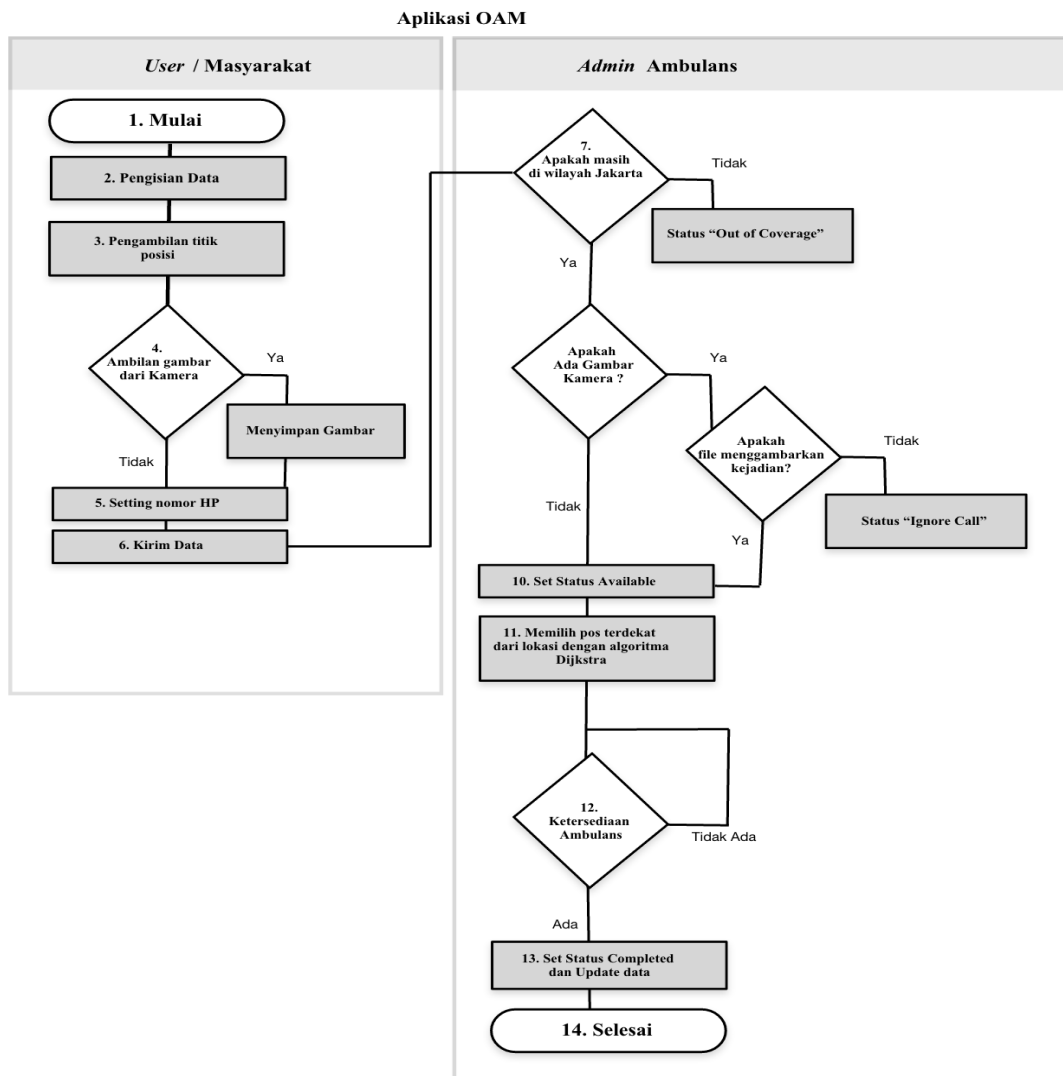
Use case diagram yang digunakan pada dapat aplikasi OAM terlihat pada gambar 3.1. Berikut pada Tabel 3.1 merupakan deskripsi dari *use case diagram* aplikasi OAM (gambar 3.1) :

Tabel 3.1 Deskripsi Use Case diagram Aplikasi OAM

Nama Use Case	Deskripsi
Melihat Informasi Aplikasi	Pada <i>use case</i> ini menjelaskan bahwa pengguna dapat membuka menu informasi aplikasi mengenai ketentuan penggunaan dan kebijakan penggunaan aplikasi <i>Order Ambulance Online</i> .
Setting Akun	Pada <i>use case</i> ini menjelaskan bahwa pengguna membuka menu setting <i>akun</i> mendaftarkan <i>save</i> nomor hp <i>user</i> , lalu OAM akan memverifikasinya ,untuk pendataan dan tersimpan untuk pemesanan ambulans ke depan.
Keluar Aplikasi	<i>Use case</i> ini menjelaskan bahwa <i>user</i> dapat keluar dari aplikasi
Pesan Ambulans	<i>Use case</i> ini menjelaskan bahwa <i>user</i> dapat memesan ambulans melalui menu ini dengan cara memasukkan beberapa informasi yang dibutuhkan.

Nama Use Case	Deskripsi
<i>Log In</i>	<i>Use case ini menjelaskan bahwa admin dapat masuk ke web based application OAM dengan cara log in terlebih dahulu.</i>
Melihat Laporan Pemesanan Ambulans	<i>Use case ini menjelaskan bahwa admin dapat melihat laporan pemesanan ambulans di web based application OAM</i>
Update Status Pemesanan Ambulans	<i>Use Case ini menjelaskan user dapat melakukan update status pada pemesanan ambulans di web based application OAM. Beberapa jenis statusnya adalah Available, Out of Coverage, Ignore Call, dan Completed.</i>
Menentukan Pos Ambulans Terdekat	<i>Use Case ini menjelaskan admin dapat menentukan pos ambulans terdekat di web based application OAM menggunakan algoritma Dijkstra</i>
Mengirimkan Data / Menerima Data	<i>Use Case ini menjelaskan user dapat mengirimkan data / admin dapat menerima data sewaktu pemesanan ambulans terjadi.</i>

b. Activity Diagram



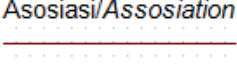
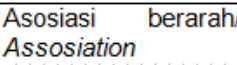
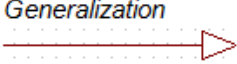
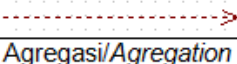
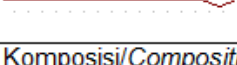


Gambar 3.2 Activity Diagram Aplikasi OAM

Activity diagram yang digunakan pada dapat aplikasi OAM terlihat pada gambar 3.2.

c. Class Diagram

Class Diagram merupakan diagram yang menggambarkan sudut sistem dari struktur dan definisi *class* serta hubungannya (*relationship*). *Class* memiliki 3 area pokok, yaitu nama, atribut, dan operasi. Nama berfungsi untuk memberikan identitas pada sebuah *class*. Atribut berfungsi untuk memberikan karakteristik pada data yang dimiliki suatu objek di dalam *class*. Operasi berfungsi untuk memberikan sebuah fungsi ke sebuah objek. *Class Diagram* memiliki beberapa simbol yang menjelaskan hubungan (*relationship*) antar *class* yang digunakan pada *class diagram*. Perhatikan tabel 3.2 untuk penjelasan simbol *class diagram*.

Tabel 3.2 Deskripsi simbol-simbol *Class Diagram*

Simbol	Keterangan
	Hubungan statis antar kelas. Asosiasi menggambarkan kelas yang memiliki atribut berupa kelas lain, atau kelas yang harus mengetahui eksistensi kelas lain. Asosiasi biasanya disertai dengan <i>multiplicity</i>
	Asosiasi dengan makna kelas yang satu digunakan oleh kelas yang lain. Asosiasi berarah juga biasanya disertai dengan <i>multiplicity</i>
	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum-khusus) atau untuk menyatakan hubungan <i>inheritance</i> .
	Relasi antar kelas dengan makna kebergantungan antar kelas.
	Hubungan yang menyatakan bahwa suatu kelas menjadi atribut bagi kelas lain
	Bentuk khusus dari agregasi dimana kelas yang menjadi bagian diciptakan setelah kelas menjadi <i>whole</i> dibuat. Misal kelas <i>whole</i> dihapus, maka kelas yang menjadi <i>part</i> ikut musnah
	Hubungan antar kelas dimana sebuah kelas memiliki keharusan untuk mengikuti aturan yang ditetapkan oleh kelas lainnya.

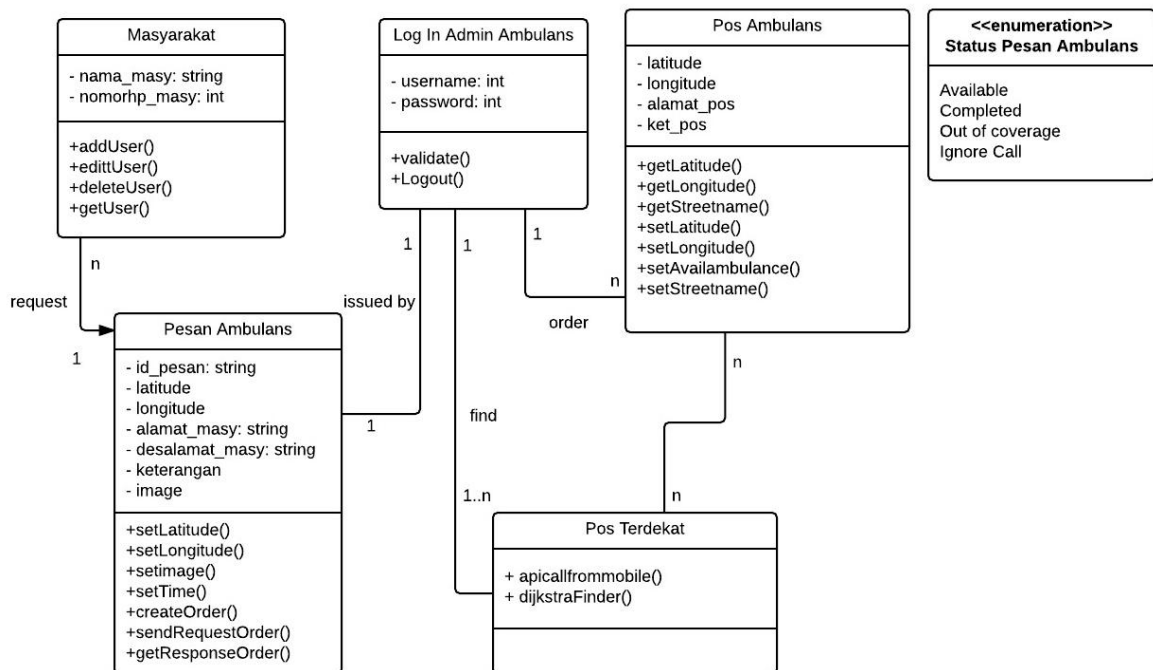
Class Diagram memiliki *multiplicity* / jumlah banyaknya objek sebuah *class* yang berelasi dengan objek dari *class* lain. Perhatikan tabel 3.3 untuk penjelasan deskripsi *multiplicity* pada *class diagram*.

Tabel 3.3 Deskripsi *Multiplicity* pada *Class Diagram*

Nilai Kardinalitas	Arti
0...1	Nol atau satu
1	Hanya satu
0...*	Nol atau lebih
1...*	Satu atau lebih
n	Hanya n (dengan n>1)
0...n	Nol sampai n (dengan n>1)
1...n	Satu sampai n (dengan n>1)

Class Diagram pada aplikasi OAM memiliki beberapa *class* yaitu *class* masyarakat, *class* pesan ambulans, *class* log in admin, *class* pos ambulans, *class* pos terdekat dan *class* status pemesanan. *Class* masyarakat memiliki beberapa atribut *private* untuk data nama dan nomor hp masyarakat. *Class* masyarakat memiliki hubungan asosiasi berarah (*directed association*) dengan *class* pesan ambulans. *Class* pesan ambulans memiliki beberapa atribut *private* untuk data id pemesanan, latitude, longitude, keterangan keadaan, kamera, alamat masyarakat serta deskripsinya. *Class* pesan ambulans memiliki hubungan dengan *class* log in admin. *Class* log in membantu merekap data pemesanan (*issued*) yang telah di *request* oleh masyarakat. *Class* log in admin memiliki beberapa atribut *private* berupa *username* dan *password*. *Class* log in admin

memiliki hubungan dengan *class* pos ambulans dan *class* pos terdekat. *Class* pos terdekat memiliki atribut *public* untuk data pencarian API dan algoritma Dijkstra. *Class* log in admin mencari pos terdekat menggunakan algoritma Dijkstra dari *class* pos terdekat. Sementara *Class* pos ambulans memiliki atribut *private* berupa data latitude, longitude, alamat pos dan keterangan alamatnya. *Class Diagram* yang digunakan pada dapat aplikasi *order ambulance mobile* (OAM), terlihat pada gambar 3.3.

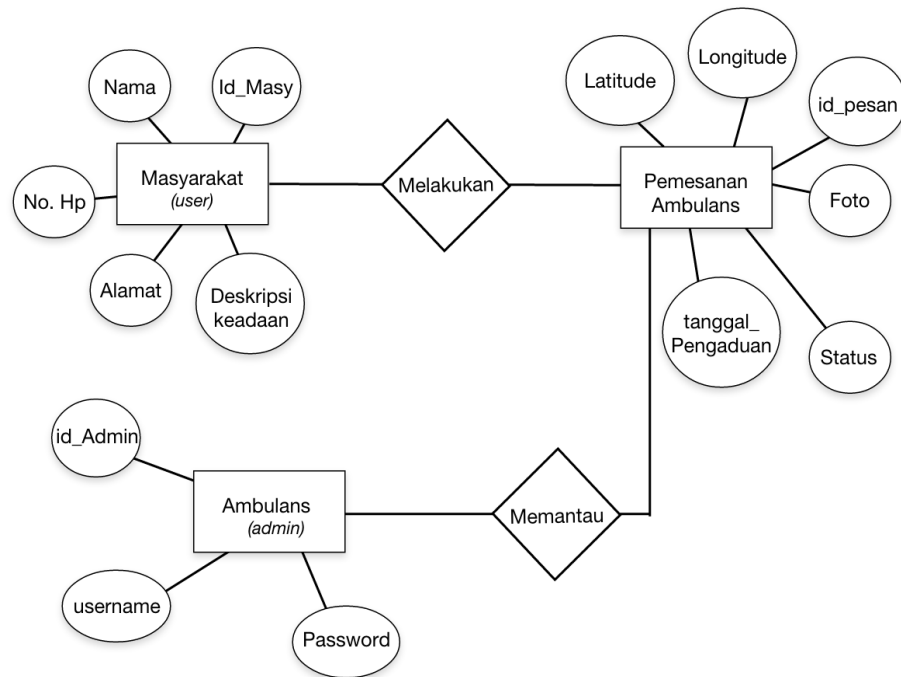


Gambar 3.3 Class Diagram Aplikasi OAM

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. Pada gambar 3.3 dapat dilihat

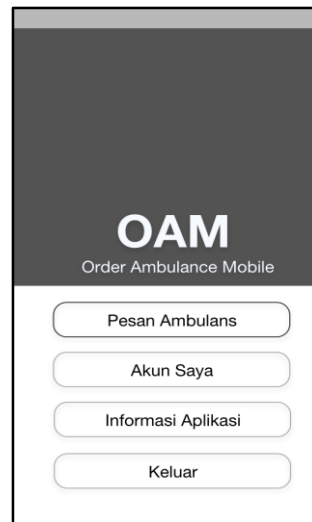
bahwa sistem ini memiliki 3 Table yang diantaranya adalah Tabel Masyarakat, Tabel Pemesanan dan Tabel Admin



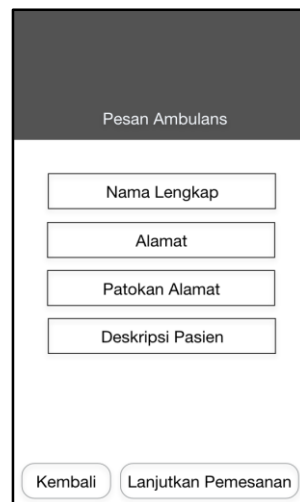
Gambar 3.4 Entity Relationship Diagram (ERD) pada OAM

e. Interface aplikasi

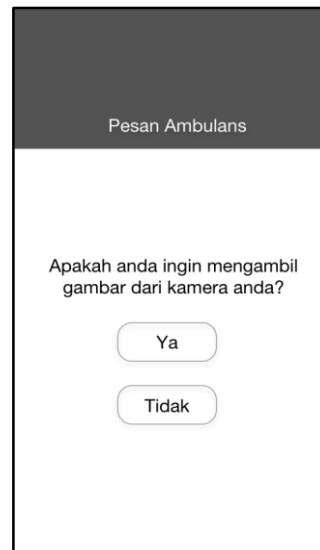
Beberapa contoh tampilan *interface* aplikasi OAM adalah sebagai berikut.



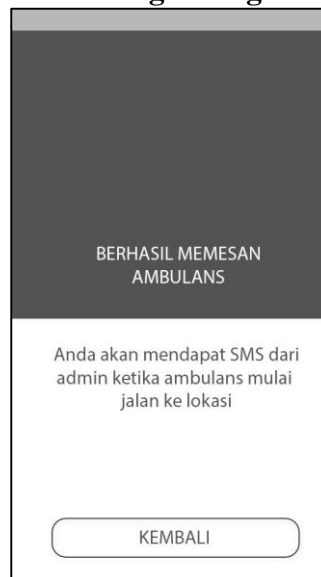
Gambar 3.5 Perancangan Antarmuka Pemesanan pada Aplikasi OAM Bagian Satu



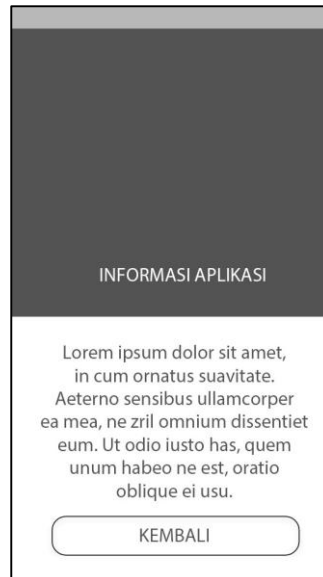
Gambar 3.6 Perancangan Antarmuka Pemesanan pada Aplikasi OAM Bagian Dua



Gambar 3.7 Perancangan Antarmuka Pemesanan pada Aplikasi OAM Bagian Tiga



Gambar 3.8 Perancangan Antarmuka Pemesanan pada Aplikasi OAM Bagian Empat



Gambar 3.9 Perancangan Antarmuka Informasi Aplikasi pada Aplikasi OAM



Gambar 3.10 Perancangan Antarmuka Akun Saya pada Aplikasi OAM

3) Fase Pengembangan Aplikasi

Fase ini merupakan fase pembuatan aplikasi yang telah di rancang pada fase - fase sebelumnya. Pada fase ini ada beberapa spesifikasi perangkat yang dibutuhkan dalam pengembangan aplikasi, yaitu :

A. Kebutuhan Perangkat Keras

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

Smartphone Android (Client)

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

Komputer (Server)

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

B. Kebutuhan Perangkat Lunak

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

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

4) Fase *Prototype* Aplikasi

Pada fase ini akan dibuat fungsi - fungsi yang akan dijalankan pada aplikasi OAM. Aplikasi akan kembali diuji pada fase selanjutnya yaitu pengujian aplikasi.

5) Fase Pengujian Aplikasi

Fase pengujian aplikasi membantu mendeteksi kesalahan aplikasi. Evaluasi kebenaran dan *running* aplikasi diujikan pada fase ini. Pada tahap evaluasi ini, pengujian dilakukan dengan mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak, atau disebut juga *black box testing*,

dan menguji cara kerja perangkat lunak untuk menganalisis kode-kode program seperti jalur logika dan seluruh stuktur data, atau disebut juga *white box testing*.

6) Fase Penyebaran Aplikasi

Fase penyebaran aplikasi dilakukan untuk memperoleh respon atas hasil aplikasi yang diciptakan. Apabila ada *bugs* yang masih terlewatkan atau belum ditemukan, kemungkinannya sangat besar akan ditemukan pada fase ini. Fase ini dapat dilakukan dengan cara penyebaran manual dari *user* ke *user* ataupun melalui website dan cara lainnya.

7) Fase *Maintenance*

Fase *maintenance* merupakan fase tahap lanjutan dari fase penyebaran aplikasi. Fase penyebaran aplikasi akan menghasilkan *feedback* / respon dari *user* yang telah mencoba aplikasi yang dibuat. Hasil dari fase penyebaran aplikasi akan digunakan untuk mengembangkan aplikasi pada fase *maintenance*.

8) Pembuatan Laporan

Pembuatan laporan penelitian aplikasi merupakan tahap terakhir penelitian ini. Prosesnya adalah mengumpulkan hasil-hasil yang di dapatkan pada penelitian dan membuat konklusi serta dokumentasi dalam bentuk tulisan.

3.3 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.

3.4 Objek Penelitian

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

3.5 Rencana Kegiatan Penelitian

Adapun rencana kegiatan penelitian ini, rencana kegiatan dapat dilihat pada Tabel 3.3

Tabel 3.3 Rencana Kegiatan Penelitian

		Nov 2015				Des 2015		Jan 2016				Feb 2016				Mar 2016				Apr 2016				Mei 2016				Juni 2016				Juli 2016				Ags 2016			
No	Jenis Kegiatan	1	2	3	4	2		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	2	3	4	2	3	4	1	2	3	4						
1	Menentukan topik																																						
2	Eksplorasi topik																																						
3	Studi Literatur																																						
4	Survei																																						
5	Menyusun Proposal																																						
6	Sidang Proposal																																						
7	Implementasi dan Evaluasi																																						
8	Menyusun Laporan TA																																						
9	Sidang TA																																						

DAFTAR PUSTAKA

- Abidin, H. Z. (2007). *Perkembangan Sistem dan Aplikasi GPS dan GLONASS*. Institut Teknologi Bandung.
- 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.
- 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/gtepwwiztAHkk3CAVHzylUJ/India-wakes-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*. Universitas Syarif Hidayatullah, Jakarta.

- Google Code. (t.th) *Android Anatomy and Physiology*. Google Code.
- 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*. Universitas Andalas, Padang.
- Lo, C. P. and A. K. W. Yeung. (2007). *Concepts and Techniques of Geographic Information Systems*. Pearson Education Inc. 532 pp.
- Leica Geosystem. (2005) *Introduction to GPS (Global Positioning System) Version 1.0 English*. University of Texas at Dallas
- Lubis, H. S. (2009). *Perbandingan Algoritma Greedy dan Dijkstra untuk Menentukan Lintasan Terpendek*. USU Repository
- Majewski, B. (2006). *Geocoding at last!*. Dipetik April 2015, dari Maps API Blog: <http://googlemapsapi.blogspot.com/2006/06/geocoding-at-last.html>
- 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
- Maghzan, K., & Jani, H. M. (2013). *A Rewiew and Evaluations Of Shortest Path Algorithms*. *International Journal of Scientific & Technology Research* Volume 2, Issue 6, June 2013.
- Maulana, A. (2015). *Jumlah pengguna Internet Indonesia Capai 88,1 Juta*. Liputan6 Tekno.
- NHS England. (2014). *Ambulance Quality Indicators Data 2014 - 15*.
- Rajabidfard, A. & Williamson, I.P. (2000). *Spatial Data Infrastructures :*

- Concept, SDI Hierarchy and Future Directions.*
- The University of Melbourne, Australia.
- Reverso Dictionary. (2000). *Collin Dictionary : False*. Reverso Dictionary.
- Riyanto. (2010). *Sistem Informasi Geografis berbasis Mobile*.
Gava Media, Yogyakarta.
- Sapic, T. (t.th). *Lecture 1 : Vector and Raster data models*. Faculty of Natural
Resources Management, Lakehead University. Kanada.
- Safaat, N. (2012). *Pemrograman Aplikasi Mobile Smartphone dan Tablet PC
Berbasis Android*. Penerbit Informatika, Bandung.
- Sanan, S., Jain, L., & Kappo, B. (2013). *Shortest Path Algorithm*.
*International Journal of Application or Innovation in Engineering &
Management (IJAIEM) Volume 2 Issue 7*.
- Saputro, Stevian S. (2013). *Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta
Wisata Di Kota 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>
- Seeber, G. (1993). *Satellite Geodesy, Foundations, Methods, and Applications*.
Walter de Gruyter, Berlin 1993.
- Statista. (2015). *Number of internet users in Indonesia from 2014 to 2019
(in millions)*.
- Sturgeon, P. (2011). *Geocoding API's Compared*.
- Somani, A.K. (2005). *Survivability and Traffic Grooming in WDM Optical Networks*.

- Cambridge University Press, Cambridge, Inggris.
- Teske, D. (2014). *Geocoder Accuracy Ranking. In Process Design for Natural Scientists: An Agile Model-Driven Approach (pp. 161-174).*
Berlin Heidelberg: Springer.
- 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.*
- Oktavia, D. (t.th). *Informasi Geografis dan Informasi Keruangan.*
Universitas Gunadarma
- Munir, R. (2005). *Matematika Diskrit.* Informatika bandung, Bandung.
- Novandi, R. A. D. (2007). *Perbandingan Algoritma Dijkstra dan Algoritma Floyd-Warshall dalam Penentuan Lintasan Terpendek (Single Pair Shortest Path).* STEI ITB.
- 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 November, Surabaya.
- Puspita, Y. (2009). *Penggunaan Arcview Gis 3.3 pada Perancangan Aplikasi Sistem Informasi Geografis Lokasi Sekolah di Wilayah Kota Bogor*
Universitas Gunadarma.
- Putra, A. T. (2015). *Jenis Data Dalam Sistem Informasi Geografis.*
Universitas Andalas.

- Prahasta, E. (2014). *Sistem Informasi Geografis Konsep-Konsep Dasar (Perspektif Geodesi & Geomatika) Edisi Revisi*. Bandung : Informatika Bandung.
- Vithani, T., & Kumar, A. (2014). Modeling the Mobile Application Development Lifecycle. *Proceedings of the International MultiConference of Engineers and Computer Scientists , I*. Hong Kong.
- Vu, P. (2003). *IDG Ventures Vietnam : Indonesia Mobile Game Industry*.
- Winardi. (2006). *Penentuan Posisi Dengan GPS Untuk Survei Terumbu Karang Jakarta*. Puslit Oseanografi, Lipi.
- Wells, D.E., N. Beck, D. Delikaraoglou, A. Kleusberg, E.J. Krawsky, G. Lachapelle, R.B. Langley, M. Nakiboglu, K.P. Schwarz, J.M. Tranquilla, P. Vanicek (1986). *Guide to GPS Positioning*. Canadian GPS Associates, Fredericton. Canada.
- 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.
- Quality Watch UK. (2014). *Ambulance Response Time*.