SKRIPSI

PENERAPAN LOCATION BASED SERVICE DAN GRAF DALAM APLIKASI PENUTUPAN JALAN

Disusun Oleh :

ALFIAN ZULFIKAR

DBC 116 028

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKA RAYA

2020

PENERAPAN LOCATION BASED SERVICE DAN GRAF DALAM APLIKASI PENUTUPAN JALAN

SKRIPSI

Sebagai salah satu syarat untuk menyelesaikan Program Strata-1 pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya

Oleh

ALFIAN ZULFIKAR

DBC 116 028

Telah dipertahankan didepan tim penguji, pada :

Hari/Tanggal : Senin, 6 Juli 2020

Waktu : 13.00-14.30 WIB

1. SHERLY CHRISTINA, S.Kom,. M.Kom. NIP. 19810929 200604 2 001 :.....(Ketua)
2. NOVERA KRISTIANTI, S.T., M.T. NIP. 0016119301 :.....(Anggota)
3. NOVA NOOR KAMALA SARI, ST.,M.Kom NIP. 19890407 201504 2 004 :.....(Anggota)
4. VIKTOR H. PRANATAWIJAYA, ST., MT NIP. 19810606 200501 1 001 :.....(Anggota)
5. PUTU BAGUS A.A.P,ST.,M.Kom NIP. 19891022 201504 1 001 :.....(Anggota)

Mengetahui :

Fakultas Teknik
Universitas Palangka Raya
Dekan,

Ir. WALUYO NUSWANTORO, M.T.
NIP. 19651119 199302 1 001

Jurusan / Program Studi Teknik Informatika
Fakultas Teknik Universitas Palangka Raya
Ketua Jurusan,

ABERTUN SAGIT SAHAY, S.T., M.Eng
NIP. 19751212 200312 1 002

PENERAPAN LOCATION BASED SERVICE DAN GRAF DALAM APLIKASI PENUTUPAN JALAN

SKRIPSI

Sebagai salah satu syarat untuk menyelesaikan Program Strata-1
pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya

Oleh :

ALFIAN ZULFIKAR

NIM. DBC 116 028

Disetujui untuk diajukan dalam Ujian Skripsi

Pembimbing I	Pembimbing II
VIKTOR HANDRIANUS P., ST., MT	**PUTU BAGUS A.A.P, ST., M.Kom**
NIP. 19810606 200501 1 001	**NIP. 19891022 201504 1 001**

JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS PALANGKA RAYA

2020

PERNYATAAN

Dengan ini saya menyatakan dengan sebenar - benarnya bahwa dalam Skripsi ini tidak terdapat karya ilmiah yang pernah diajukan untuk memperoleh gelar kesarjanaan disuatu Perguruan Tinggi, serta tidak terdapat karya ilmiah atau pendapat yang pernah ditulis atau diterbitkan orang lain, kecuali secara tertulis diacu dalam Skripsi ini dan disebutkan dalam Tinjauan Pustaka.

Palangka Raya, Juli 2020

ALFIAN ZULFIKAR
DBC 116 028

RIWAYAT PENYUSUN

Data Diri

Nama	: Alfian Zulfikar
NIM	: DBC 116 028
Fakultas	: Teknik
Jurusan/Program Studi	: Teknik Informatika
Jenjang	: Strata 1 (S-1)
Jenis Kelamin	: Laki-laki
Tempat, Tanggal Lahir	: Tanjung Mas, 8 November 1997
Agama	: Islam
Status dalam Keluarga	: Anak Kandung
Anak ke -	: 2
Alamat	: Jl. Bukit Indah
No. Telpon/HP	: +6287742351592

Data Orang Tua

Nama Ayah	: Martoyo
Pekerjaan Ayah	: Karyawan swasta
Nama Ibu	: Dewi
Pekerjaan Ibu	: Ibu rumah tangga
Alamat Orang Tua	: Jl. Cendrawasih no.198 Perumnas Pembina Sampit
No. Telpon/HP	: +6285787306774

Riwayat Pendidikan

SD	: SDN 4 Mentawa Baru Hilir	(Tahun Lulus 2010)
SMP	: SMPN 1 Sampit	(Tahun Lulus 2013)
SMA	: SMAN 1 Sampit	(Tahun Lulus 2016)

Palangka Raya, Juli 2020

ALFIAN ZULFIKAR
DBC 116 028

HALAMAN PERSEMBAHAN

Dengan mengucap Alhamdulillah, kupersembahkan karya kecilku ini untuk orang-orang yang kusayangi:

1. Bapak Ibu tercinta, dua sosok yang tak pernah lelah memberikan dukungan dalam bentuk materi dan moral. Semoga keberhasilanku menjadi sebab Bapak dan Ibu mendapatkan Surga-nya Allah *subhanahu wa ta'ala*. I love you Dad, I love you Mom.
2. Indriana Martha Dewi, kakak tercinta yang selalu memperhatikan dan mengingatkan adiknya setiap waktu selama masa kuliah. Aku berhasil mbak.
3. Anis Hidayati dan keluarga, sepupuku yang pernah memberikan tempat tinggal yang sangat layak, memberi makan dan meminjamkan kendaraannya selama 2 semester perkuliahan.
4. *Waliyul Amri*, Pemerintah Indonesia yang terhormat, yang telah memberikan program beasiswa Bidikmisi, sehingga aku tidak perlu pusing memikirkan biaya kuliah. Semoga Allah *subhanhu wa ta'ala* selalu membimbing kalian.
5. Sahabat-sahabat pengajian sunnah Palangka Raya, Junaidi, Harits, Ahmadi, Fajri dan lainnya, yang menjadi lingkungan terbaik bagiku untuk selalu mengingat Allah *subhanahu wa ta'ala* dalam lingkungan perkuliahan yang penuh dengan hal-hal yang sangat mungkin menjerumuskan diri ini dalam kemaksiatan.
6. Ihwan dan Rizal, teman satu atap selama kurang lebih 3 tahun, yang saling menjaga dan melengkapi kebutuhan satu sama lain.
7. Sahabat-sahabat seperjuangan, mahasiswa jurusan Teknik Informatika, Fakultas Teknik, Universitas Palangka Raya angkatan 2016. Kita sudah bersama-sama memulai perjuangan hingga berada pada titik ini. I'll miss you guys. Semangat!
8. Dan untuk kamu.

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah *Subhanahu Wata'ala*, tuhan yang maha esa. Karena atas berkat rahmat dan karunia-Nya penulis dapat menyelesaikan skripsi dengan judul "*Penerapan Location Based Service dan Graf dalam Aplikasi Penutupan Jalan*". Dengan telah rampungnya skripsi ini, penulis mengucapkan terima kasih kepada dosen pembimbing skripsi, bapak Viktor Handrianus Pranatawijaya, S.T., M.Eng. dan bapak Putu Bagus A.A.P, ST., M.Kom yang telah memberikan bimbingan kepda penulis dalam menyelesaikan skripsi ini, serta rekan-rekan yang telah memberikan dukungan.

Penulis menyadari akan segala kekurangan yang ada pada Skripsi ini. Maka dari itu hendaknya pembaca dapat memberikan tanggapan dan saran yang membangun agar kedepannya bisa menjadi lebih baik lagi. Penulis sangat berharap Skripsi ini dapat berguna dan bermanfaat bagi diri penulis sendiri, serta lebih luas lagi bagi para pembaca dalam rangka menambah wawasan serta pengetahuan pembaca mengenai metode, prinsip kerja, dan pengimplementasian program yang dibangun dari penelitian ini.

Palangka Raya, Juli 2020

Penulis

PENERAPAN LOCATION BASED SERVICE DAN GRAF PADA APLIKASI PENUTUPAN JALAN

Alfian Zulfikar DBC 116 028
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Palangka Raya
Kampus Tunjung Nyaho Jl. Yos Sudarso Palangka Raya 73112
alfian.zulfikar789@gmail.com

ABSTRAK

Penutupan jalan merupakan isu yang kerap terjadi di kota besar. Yang sering terjadi adalah penutupan jalan tidak disertai dengan adanya informasi yang valid bagi masyarakat, sehingga masyarakat tidak mengetahui adanya penutupan jalan. Sehingga pada penelitian ini, akan dilakukan penerapan teknologi *Location Based Service* (LBS) dan struktur data Graf dalam pembuatan sebuah aplikasi yang dapat digunakan masyarakat untuk membuat dan mendapatkan informasi penutupan jalan. Tidak hanya itu, aplikasi yang dibuat juga disertai dengan fitur *request* rute alternatif, sehingga masyarakat bisa mendapatkan rute alternatif sebelum beraktifitas.

Aplikasi dikembangkan dengan menggunakan metodologi *Extreme Programming*. Pengembangan aplikasi dimulai dengan pendefinisian kebutuhan user dengan *user stories*. Kemudian membuat *planning* untuk menentukan aspek sistem yang akan dikembangkan serta kriteria pengujian yang akan dilakukan. Dilanjutkan dengan pembuatan desain, implementasi program dan pengujian program. Pengembangan aplikasi dilakukan sebanyak 2 iterasi, dikarenakan adanya penambahan kebutuhan pada *user stories*, yaitu fitur untuk mendapatkan peringatan penutupan jalan pada radius tertentu. Aplikasi yang dibuat berbasis *mobile* dan *website*. Aplikasi mobile digunakan oleh user public atau masyarakat, sedangkan aplikasi yang berbasis website digunakan oleh operator sebagai pengelola informasi.

Hasil dari penelitian ini, LBS berhasil diterapkan untuk menentukan titik penutupan jalan serta membuat peringatan penutupan jalan dalam bentuk *push notification* menggunakan fungsi *Geofence*. Notifikasi muncul ketika perangkat berada pada radius kurang dari 100 meter dari titik penutupan jalan, baik aplikasi dalam keadaan aktif atau berjalan dilatar belakang. Sementara teori Graf berhasil diterapkan untuk membuat rute alternatif dengan memanfaatkan titik koordinat jalan sebagai node penyusun Graf. Pemrosesan rute dilakukan dengan *rules* yang dibuat dari hasil modifikasi algoritma *Breadth First Search*, yaitu dengan menambahkan proses pemeriksaan titik hambatan. Sehingga rute yang di-*generate* berhasil menghindari titik penutupan jalan. Berbeda dengan hasil pengujian dengan menerapkan algoritma *Breadth First Search* original, dimana rute yang dihasilkan tetap melewati titik penutupan.

Kata kunci : Penutupan Jalan, *Location Based Service*, Graf, *Breadth First Search*, *Extreme Programming*.

IMPLEMENTATION OF LOCATION BASED SERVIDE AND GRAPH IN THE ROAD CLOSURE APPLICATION

Alfian Zulfikar DBC 116 028
Jurusan Teknik Informatika, Fakultas Teknik, Universitas Palangka Raya
Kampus Tunjung Nyaho Jl. Yos Sudarso Palangka Raya 73112
alfian.zulfikar789@gmail.com

ABSTRACT

Road closure is an issue that often occurs in most big cities. What often happens is that road closures are not equipped by valid information for people, so people are not aware of road closures. So in this research, the implementation of Location Based Service (LBS) technology and Graph data structure will be carried out in making an application which can be used by people to create and get road closure information. Furthermore, the application is also equipped by an alternative route request feature, so that people can get alternative route before they start their activities.

The application was developed using methodology of Extreme Programming. Application development starts with defining user needs with user stories. Then make planning to determine some aspects of the system to be developed and the testing criteria. Followed by design, program implementation and program testing. Application development is done in 2 iterations, due to additional user stories, which is a feature to get road closure warnings at a certain radius. Application made based on mobile and website. Mobile-based applications are used by public users, while website-based applications are used by operators as information manager.

Based on the results of this research, LBS was successfully implemented to determine the road closure point and make road closure warnings in the form of push notifications by using Geofence. Notifications appear when the device position is less than 100 meters from the road closure point, either when the application is active or running in the background. While Graph theory was successfully applied to create alternative routes by utilizing road coordinate points as nodes which build the Graph. Route processing is done by using rules that made from the modification of Breadth First Search algorithm, by adding resistance checking process. So that the generated route successfully avoids the closure point. Different result given in the testing with original Breadth First Search algorithm, that produced route that still through the closure point.

Keywords : Road closure, Location Based Service, Graph, Breadth First Search, Extreme Programming.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Setiap kota memiliki infrastrukur yang menunjang aktivitas masyarakatnya. Tanpa infrastruktur yang memadai, masyarakat akan kesulitan dalam menjalankan aktivitas sehari-hari. Jalan merupakan salah satu infrastruktur yang sangat berpengaruh terhadap aktivitas masyarakat. Jalan akan menghubungkan suatu tempat dengan tempat lainnya, sehingga masyarakat dapat mengambil rute sebuah jalan untuk menuju ke suatu tempat. Dengan meningkatnya konektifitas, maka akan berdampak pula pada semakin mudahnya pemenuhan kebutuhan masyarakat.

Masalah infrastruktur jalan yang kerap terjadi di berbagai kota adalah ditutupnya ruas jalan tertentu tanpa adanya pemberitahuan atau pengumuman sebelumnya. Penutupan ruas jalan tersebut diakibatkan oleh banyak hal, seperti perbaikan jalan, jalan tergenang air, terjadinya kecelakaan lalu lintas, hingga yang sering terjadi adalah penggunaan badan jalan untuk *event* tertentu seperti *event* keluarga, keagamaan dan kebudayaan. Tentu saja penutupan jalan ini berdampak pada pengguna jalan yang pada saat bersamaan perlu menggunakan jalan tersebut untuk menuju suatu tempat. Sehingga bagi pengguna jalan yang tidak mengetahui adanya penutupan jalan dan terlanjur mengambil jalan tersebut, mereka harus memutar arah dan mencari alternatif jalan lain. Hal Ini tentu sangat menguras waktu, terutama bagi pengguna jalan yang tergesa-gesa dan tidak mengira jika jalan tersebut ditutup.

Pemanfaatan teknologi *Location Based Service* merupakan salah cara dalam menyelesaikan masalah ini. *Location Based Service* atau yang dapat diartikan sebagai layanan berbasis lokasi, merupakan sebuah teknologi yang berlandaskan pada pemanfaatan data lokasi geografis suatu perangkat, yaitu berupa koordinat garis lintang dan garis bujur. Berkaitan dengan permasalahan ini, LBS dapat dimanfaatkan sebagai cara untuk membuat informasi adanya penutupan jalan

dengan mengambil data lokasi tempat terjadinya penutupan jalan, yang kemudian data diolah menjadi informasi yang siap dipublikasikan kepada masyarakat.

Informasi mengenai penutupan jalan tersebut juga menjadi lebih baik jika diberikan rute alternatif yang ditawarkan kepada pengguna. Sehingga untuk memenuhi kebutuhan tersebut, pada penelitian ini pula akan dilakukan pengolahan data lokasi dengan membuat struktur data Graf untuk merepresentasikan rute jalan yang kemudian dapat dimanipulasi untuk mendapatkan rute alternatif guna menghindari adanya penutupan jalan. Berdasarkan latar belakang yang sudah dipaparkan, diangkatlah sebuah judul penelitian skripsi yaitu "**PENERAPAN LOCATION BASED SERVICE DAN GRAF DALAM APLIKASI PENUTUPAN JALAN**".

1.2. RUMUSAN MASALAH

Berdasarkan paparan latar belakang dari penelitian ini, rumusan masalah yang didapatkan adalah bagaimana penerapan teknologi *Location Based Service* dalam pengembangan aplikasi penutupan jalan, serta bagaimana penerapan teori Graf untuk merepresentasikan jalan pada aplikasi yang dibuat.

1.3. BATASAN MASALAH

Adapun batasan masalah dari penelitian ini adalah sebagai berikut.

1. Pengguna dari aplikasi ini adalah *operator* dan *public*.
2. Terdapat dua aplikasi yang dibuat, yaitu berbasis *website* untuk *operator* dan berbasis *mobile* untuk masyarakat.
3. Aplikasi yang berbasis *mobile* berjalan pada sistem operasi Android.
4. Informasi penutupan jalan berupa deskripsi, koordinat lokasi waktu penutupan jalan dan level penutupan jalan.
5. Dalam membuat representasi jalan menggunakan struktur data Graf.
6. Penentuan rute alternatif menggunakan *rules* yang dibuat dengan memodifikasi algoritma *Breadth First Search* berupa penambahan proses pengecekan node hambatan.

7. Aplikasi dibuat menggunakan bahasa pemrograman Java dan *framework* Codeigniter, serta menggunakan MySQL sebagai DBMS.
8. Fitur dari aplikasi yang dibuat adalah sebagai berikut.
    a. Operator
        1) Kelola informasi (*approve*, hapus).
        2) Kelola jalan (menambahkan persimpangan).
        3) Kelola public (hapus data *public*).
        4) Kelola akun (ganti *password*)
    b. Public
        1) Buat informasi.
        2) Lihat informasi.
        3) Lihat lokasi penutupan jalan via *Maps*.
        4) Berbagi informasi via aplikasi eksternal.
        5) *Request* rute alternatif.
9. Cakupan untuk pemrosesan rute alternatif meliputi sebagian wilayah di kecamatan Jekan Raya kota Palangka Raya, dengan batasan sebagai berikut.
    1) Bundaran Besar
    2) Persimpangan Jl. Yos Sudarso / Jl. Bukit Kaminting
    3) Persimpangan Jl. Bukit Kaminting / Jl. Garuda
    4) Persimpangan Jl. Garuda / Jl. Tjilik Riwut

1.4. TUJUAN

Tujuan dari penelitian ini adalah dapat menerapkan teknologi *Location Based Service* dan teori Graf dalam pengembangan aplikasi penutupan jalan, sehingga pengguna dapat berbagi dan mendapatkan informasi mengenai penutupan jalan serta mendapatkan rute alternatif.

1.5. MANFAAT

Manfaat dari penelitian ini adalah terciptanya aplikasi yang dapat digunakan masyarakat untuk berkontributsi dalam membuat informasi mengenai penutupan jalan, sehingga masyarakat atau pengguna jalan lain bisa mendapatkan informasi

terkini mengenai penutupan jalan. Selain itu, penelitian ini juga bermanfaat untuk mengetahui hasil dari implementasi teori Graf dalam membuat representasi jalan.

1.6. SISTEMATIKA PENULISAN

Dalam pembuatan laporan akhir skripsi, penulis menggunakan sistematika penulisan yang terdiri dari 5 bab, yaitu sebagai berikut.

BAB I : PENDAHULUAN

Pada bab ini, diuraikan mengenai latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat, sistematika penulisan dan jadwal kegiatan.

BAB II : LANDASAN TEORI

Landasan teori berisi teori yang digunakan sebagai landasan penelitian yang didapatkan dari penelitian sebelumnya atau teori dari seorang pakar. Landasan teori ini dipaparkan dalam bentuk tinjauan pustaka dan teori pendukung.

BAB III : METODOLOGI PENELITIAN

Bab ini berisi penjelasan tahap-tahap yang dilalui pada penelitian, lokasi penelitian, metode pengumpulan data, metode pengolahan data dan bagian perancangan.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini menjelaskan hasil dari penelitian yang telah dilakukan, yaitu hasil pengujian dari aplikasi yang dibuat. Pembahasan dilakukan seputar fitur-fitur dari aplikasi serta pengaplikasiannya.

BAB V : KESIMPULAN DAN SARAN

Bab ini berisi kesimpulan dan saran dari awal sampai terbentuknya aplikasi. Kesimpulan memuat jawaban atau ringkasan atas permasalahan yang di jabarkan pada rumusan masalah. Sedangkan saran berisi saran-saran yang perlu diperhatikan berdasarkan keterbatasan-keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama pengembangan perangkat lunak.

1.7. JADWAL SKRIPSI

Tabel 1.1. Jadwal Skripsi

No.	Kegiatan	Bulan dan Minggu																			
		Maret				April				Mei				Juni				Juli			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Penyusunan dan pengumpulan proposal																				
2	Planning dan desain																				
3	Implementasi dan pengujian																				
4	Pembuatan laporan akhir skripsi																				
5	Seminar hasil																				
6	Seminar akhir																				

BAB II
LANDASAN TEORI

2.1. TINJAUAN PUSTAKA

Peneletian yang dilakukan memiliki fokus pada penerapan teknologi *Location Based Service* dan teori Graf dalam pengembangan aplikasi penutupan jalan. Penulis meninjau beberapa penelitian yang pernah dilakukan berkaitan dengan topik penelitian skripsi ini. Berikut ini beberapa penelitian yang ditinjau penulis.

Nur Alam dan Mukhlis Amin (2015) dalam Jurnal Pekommas, Vol. 18 No. 2, Agustus 2015: 93 - 104 yang berjudul *Aplikasi Pemilihan Rute Alternatif Akibat Kemacetan Lalu Lintas di Kota Makassar Menggunakan Google API dan ASP.Net*, dilakukan penelitian untuk mengatasi persoalan kemacetan yang semakin parah di Kota Makassar dengan memberikan sebuah solusi aplikasi pemilihan rute alternatif untuk mengatasi kemacetan. Aplikasi ini dapat menampilkan lokasi kemacetan yang sedang terjadi beserta jalur alternatif yang memungkinkan untuk dilalui agar terhindar dari kemacetan tersebut. Rute alternatif diperoleh berdasarkan layanan yang telah disediakan oleh Google API. Penentuan titik kemacetan dilakukan melalui telepon seluler dengan aplikasi berbasis android menggunakan bahasa pemrograman eclipse. Sementara itu, aplikasi untuk mendesain rute alternatif menggunakan Active Server Pages (ASP.Net).

Taufik Hery Purwanto (2014) dalam penelitiannya yang berjudul *Pembuatan Rute Alternatif Berbasis Web-GIS Untuk Menghindari Kemacetan Lalulintas Di Kota Tangerang Selatan*, peneliti membuat basisdata dalam membuat rute alternatif menghindari kemacetan lalulintas dan membuat sistem informasi visualisasi rute berbasis Web-GIS. Implementasi algoritma pencarian rute A-star dilakukan pada proses pengolahan basisdata menggunakan sistem basisdata PostgreSQL dan PostGIS untuk data spasial, sebagai parameter impedansi digunakan panjang dari tiap ruas jalan yang akan menunjukkan jarak terpendek. Diketahui dari hasil penelitian tersebut, Web-GIS menampilkan informasi sesuai

basis data spasial, navigasi peta, pencarian rute dengan kemampuan menerima lokasi awal dan tujuan dari pengguna untuk menghindari ruas jalan yang macet.

Mira Kusmira dan Taufiqurrochman (2017) dalam penelitiannya yang berjudul *Pemanfaatan Aplikasi Graf Pada Pembuatan Jalur Angkot 05 Tasikmalaya*, melakukan penelitian untuk membuat jalur angkutan kota dengan menggunakan teori graf. Pada penelitian ini dilakukan pengujian teori Graf untuk membuat jalur angkot di Tasikmalaya dengan representasi lintasan dan sirkuit Euler.

Dari ketiga penelitian yang ditinjau, terdapat 2 penelitian tentang pemilihan rute alternatif untuk menghindari kecelakaan. Kedua penelitian ini mirip dengan penelitian yang akan dilakukan. Hanya saja dalam penelitian skripsi ini akan dibuat aplikasi untuk berbagi informasi adanya penutupan jalan dengan sebab yang lebih luas seperti yang dipaparkan pada Bab 1. Jika kedua penelitian tersebut menggunakan layanan Google API dan Web GIS untuk mendapatkan rute, maka pada penelitian ini rute didapatkan dengan membuat representasi jalan menggunakan teori Graf, sehingga untuk mendapatkan rute alternatif dilakukan dengan memanipulasi struktur Graf yang dibuat. Pengujian Graf sebagai representasi rute jalan juga dilakukan pada penelitian ketiga yang ditinjau penulis. Pada penelitian yang akan dilakukan, struktur Graf yang digunakan adalah Graf tidak berarah dengan memungkinkan adanya penambahan node baru yang menggambarkan titik asal, titik tujuan dan titik penutupan jalan sebagai hambatan.

2.2. LOCATION BASED SERVICE

Viktor Handrianus Pranatawijaya, Deddy Ronaldo, dan Farhani (2018) dalam Jurnal Teknologi Informasi, Vol. 12 No. 1, Agustus 2018: 65 - 73 yang berjudul *Penerapan Location Based Service Pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya,* LBS adalah pengiriman data dan layanan informasi dimana isi dari layanan tersebut disesuaikan dengan saat ini atau beberapa lokasi yang diproyeksikan dan konteks pengguna *mobile*. LBS didasarkan pada tiga kategori lokasi yaitu lokasi deskripsi, spatial, dan jaringan.

Aplikasi berbasis lokasi adalah salah satu yang paling diantisipasi pada segmen industri *mobile*. Aplikasi ini baru dapat diaktifkan oleh ponsel yang dilengkapi GPS. Layanan Berbasis Lokasi atau lebih dikenal dengan *Location Based Service* (LBS) istilah umum yang digunakan untuk menggambarkan teknologi yang digunakan untuk menemukan lokasi perangkat yang kita gunakan. LBS adalah layanan informasi yang dapat diakses melalui *mobile device* dengan menggunakan *mobile network*, yang dilengkapi kemampuan untuk memanfaatkan lokasi dari *mobile device* tersebut. LBS berisi sejumlah komponen termasuk informasi peta dan GIS, layanan penentuan lokasi, dan LBS subkomponen aplikasi tertentu (Amit Kushwaha, 2011).

Terdapat dua unsur utama pada LBS yaitu:

1) *Location Manager* (API *Maps*), untuk menyediakan tools/source untuk LBS, *Application Programming Interface* (API) Maps menyediakan fasilitas untuk menampilkan, memanipulasi maps/peta beserta fitur-fitur lainnya seperti tampilan satelit, *street* (jalan), maupun gabungannya. Paket ini berada pada *com.google.adroid.maps*.
2) *Location Provider* (API *Location*), untuk menyediakkan teknologi pencarian lokasi yang digunakan oleh *device*/perangkat. API *Location* berhubungan dengan data GPS (*Global Positioning System*) dan data lokasi *real-time*. API *Location* berada pada paket *android* yaitu dalam paket *android.location*. Dengan *Location Manager*, kita dapat menentukan lokasi kita saat ini, *track* gerakan/perpindahan, serta kedekatan dengan lokasi tertentu dengan mendeteksi perpindahan.

Viktor Handrianus Pranatawijaya, Deddy Ronaldo, dan Farhani (2018) dalam Jurnal Teknologi Informasi, Vol. 12 No. 1, Agustus 2018: 65 - 73 yang berjudul *Penerapan Location Based Service Pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya*, terdapat lima komponen pendukung utama dalam teknologi Layanan Berbasis Lokasi, antara lain:

1) Piranti *Mobile*, adalah salah satu komponen penting dalam LBS. Piranti ini berfungsi sebagai alat bantu (*tool*) bagi pengguna untuk meminta informasi. Hasil dari informasi yang diminta dapat berupa teks, suara, gambar dan lain

sebagainya. Piranti *mobile* yang dapat digunakan bias berupa PDA, *smartphone*, laptop. Selain itu, piranti *mobile* dapat juga berfungsi sebagai alat navigasi di kendaraan seperti halnya alat navigasi berbasis GPS.

2) Jaringan Komunikasi, Komponen ini berfungsi sebagai jalur penghubung yang dapat mengirimkan data-data yang dikirim oleh pengguna dari piranti *mobile*-nya untuk kemudian dikirimkan ke penyedia layanan dan kemudian hasil permintaan tersebut dikirimkan kembali oleh penyedia layanan kepada pengguna.
3) Komponen *Positioning* (Penunjuk Posisi/Lokasi), Setiap layanan yang diberikan oleh penyedia layanan biasanya akan berdasarkan pada posisi pengguna yang meminta layanan tersebut. Oleh karena itu diperlukan komponen yang berfungsi sebagai pengolah/pemroses yang akan menentukan posisi pengguna layanan saat itu. Posisi pengguna tersebut bisa didapatkan melalui jaringan komunikasi *mobile* atau juga menggunakan *Global Positioning System* (GPS).
4) Penyedia layanan dan aplikasi, merupakan komponen LBS yang memberikan berbagai macam layanan yang bisa digunakan oleh pengguna.
5) Penyedia data dan konten, Penyedia layanan tidak selalu menyimpan seluruh data dan informasi yang diolahnya. Karena bisa jadi berbagai macam data dan informasi yang diolah tersebut berasal dari pengembang/pihak ketiga yang memang memiliki otoritas untuk menyimpannya. Sebagai contoh basis data geografis dan lokasi bisa saja berasal dari badan-badan milik pemerintah.

2.3. TEORI GRAF

Menurut Mira Kusmira dan Taufiqurrochman (2017) dalam penelitiannya yang berjudul *Pemanfaatan Aplikasi Graf Pada Pembuatan Jalur Angkot 05 Tasikmalaya*, Graf dapat didefinisikan sebagai himpunan tidak kosong antara pasangan simpul-simpul dan sisi-sisi yang menghubungkan sepasang simpul. Himpunan simpul tidak boleh kosong, sedangkan himpunan sisi boleh kosong. Jadi suatu titik juga bisa disebut suatu graf. Graf yang hanya terdiri dari satu buah

simpul tanpa sebuah sisi pun disebut graf trivial. Berdasarkan ada tidaknya gelang atau sisi ganda pada suatu graf, graf dapat digolongkan menjadi dua jenis:

a. Graf Sederhana

Graf sederhana adalah graf yang tidak mengandung gelang maupun sisi ganda. Contoh graf sederhana direpresentasikan dengan jaringan komputer. Pada graf sederhana sisi merupakan pasangan tak terurut. Jadi sisi (u,v) sama saja dengan (v,u).

b. Graf tak Sederhana

Graf tak sederhana adalah graf yang mengandung sisi ganda atau gelang. Graf sederhana dibagi menjadi dua macam, yaitu graf ganda dan graf semu. Graf ganda adalah graf yang mengandung sisi ganda. Sedangkan graf semu adalah graf yang mengandung gelang. Sisi pada graf semu dapat terhubung ke dirinya sendiri.

Berdasarkan orientasi arah pada sisi, maka graf dbedakan menjadi dua jenis:

a. Graf Tak berarah

Graf tak berarah adalah graf yang sisinya tidak mempunyai orientasi arah. Urutan pasangan simpul pada graf beraaah tidak diperhatikan, jadi sisi (u,v) sama dengan (v,u). contoh graf tak berarah dalam kehidupan sehari hari adalah jaringan pada saluran secara dua arah.

b. Graf berarah

Graf berarah adalah graf yang setiap sisinya diberikan orientasi arah. Sisi sisinya yang berarah ini biasa disebut busur. Pada graf berarah, sisi(u,v) tidak sama dengan (v,u). untuk busur (u,v), simpul u merupakan simpul terminal. Dalam kehidupan sehari-hari, garaf berarah biasa dipakai untuk menggambarkan aliran suatu proses.

2.4. JALAN

Menurut UU RI no. 38 Tahun 2004 pasal 1 ayat (4) jalan adalah prasarana transportasi darat yang meliputi segala bagian jalan, termasuk bangunan pelengkap dan perlengkapannya yang diperuntukkan bagi lalu lintas yang berada pada permukaan tanah dan atau air, serta di atas permukaan air, kecuali jalan kereta api, jalan lori, dan jalan kabel.

a. Menutut Sistem Jaringan Jalan

1) Sistem jaringan jalan primer merupakan sistem jaringan jalan dengan pelayanan distribusi barang dan jasa untuk pengembangan semua wilayah di tingkat nasional, dengan menghubungkan semua simpul jasa distribusi yang berwujud pusat-pusat kegiatan.
2) Sistem jaringan jalan sekunder merupakan sistem jaringan jalan dengan pelayanan distribusi barang dan jasa untuk masyarakat di dalam kawasan perkotaan.

b. Menurut Fungsinya

1) Jalan arteri merupakan jalan umum yang berfungsi melayani dengan ciri perjalanan jarak jauh, kecepatan rerata tinggi, dan jumlah jalan masuk dibatasi secara berdaya guna.
2) Jalan kolektor merupakan jalan umum yang berfungsi melayani angkutan pengumpul atau pembagi dengan ciri perjalanan jarak sedang, kecepatan rerata sedang, dan jumlah jalan masuk dibatasi.
3) Jalan lokal merupakan jalan umum yang berfungsi melayani angkutan setempat dengan ciri perjalanan jarak dekat, dan kecepatan rerata rendah.
4) Jalan lingkungan merupakan jalan umum yang berfungsi melayani angkutan lingkungan dengan ciri perjalanan jarak dekat, dan kecepatan rerata rendah

c. Menurut Statusnya

1) Jalan nasional merupakan jalan arteri dan jalan kolektor dalam sistem jaringan jalan primer yang menghubungkan antar ibukota provinsi, dan jalan strategis nasional serta jalan tol.

2) Jalan provinsi merupakan jalan kolektor dalam sistem jaringan jalan primer yang menghubungkan ibukota provinsi dengan ibukota kabupaten/kota, atau antar ibukota kabupaten/kota, dan jalan strategis provinsi.
3) Jalan kabupaten merupakan jalan lokal dalam sistem jaringan jalan primer yang menghubungkan ibukota kabupaten dengan ibukota kecamatan, antar ibukota kecamatan, ibukota kabupaten dengan pusat kegiatan lokal, antar pusat kegiatan lokal, serta jalan umum dan sistem jaringan sekunder dalam wilayah kabupaten, dan jalan strategis kabupaten.
4) Jalan kota merupakan jalan umum dalam sistem jaringan jalan sekunder yang menghubungkan antar pusat pelayanan dalam kota, menghubungkan pusat pelayanan dengan persil, menghubungkan antar persil, serta menghubungkan antar pusat pemukiman yang berada di dalam kota.
5) Jalan desa merupakan jalan umum yang menghubungkan kawasan dan atau antar pemukiman di dalam desa, serta jalan lingkungan.

2.5. ALGORITMA BREADTH FIRST SEARCH

Menurut Budi Prasetiyo dan Maulidia Rahmah Hidayah (2014) dalam *Scientific Journal of Informatics* dengan judul *Penggunaan Metode Depth First Search (DFS) dan Breadth First Search (BFS) pada Strategi Game Kamen Rider Decade Versi 0.3*, *Breadth First Search* adalah suatu metode yang melakukan pencarian secara melebar yang mengunjungi simpul secara preorder yaitu mengunjungi suatu simpul kemudian mengunjungi semua simpul yang bertetangga dengan simpul tersebut dahulu. Selanjutnya, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya.

Pada algoritma BFS, simpul anak yang telah dkunjungi disimpan dalam suatu antrian. Antrian ini digunakan untuk mengacu simpul-simpul yang bertetangga dengannya yang akan dikunjungi kemudian sesuai urutan pengantrian. Beberapa keuntungan menggunakan algoritma BFS yaitu tidak akan menemui jalan buntu dan jika ada satu solusi maka BFS akan menemukannya, jika ada lebih

dari satu solusi maka solusi minimum akan ditemukan. Langkah-langkah pada algoritma BFS adalah sebagai berikut.

1) Masukan simpul ujung (akar) ke dalam antrian.
2) Ambil simpul dari awal antrian, lalu cek apakah simpul merupakan solusi.
3) Jika simpul merupakan solusi, pencarian selesai dan hasil diekmbalikan.
4) Jika simpul bukan solusi, masukan seluruh simpul yang bertetangga dengan simpul tersebut (simpul anak) ke dalam antrian.
5) Jika antrian kosong dan setiap simpul sudah dicek, pencarian selesai dan mengembalikan hasil solusi tidak ditemukan.
6) Ulangi pencarian dari langkah kedua.

2.6. UNIFIED MODELLING LANGUAGE

UML atau *Unified Modelling Language* adalah salah satu standar bahasa yang banyak digunakan di dunia industri untuk mendefinisikan requirement, membuat analisis dan desain, serta menggambarkan arsitektur dalam pemrograman berorientasi objek. Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang di berbagai negara dapat mengerti permodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodela perangkat lunak yang dapat dimengerti.

UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung. Penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataanya UML paling banyak digunakan pada metodologi berorientasi objek. Beberapa jenis diagram UML adalah sebagai berikut.

a. Use case diagram

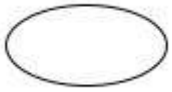
Menurut Rosa dan Shalahuddin (2016) dalam bukunya yang berjudul *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*, *use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor engan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan sapa saja yang berhak menggunakan fungsi-fungsi itu.

Syarat penamaan pada *use case* adalah nama didefinsikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

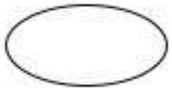
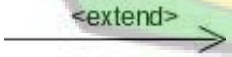
1) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2) *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut adalah simbol-simbol yang ada pada diagram *use case*:

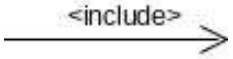
Tabel 2.1. Simbol *Use case diagram*.

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, dinyatakan dalam kata kerja diawal frase nama *use case*.

Tabel 2.1. (lanjutan).

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, dinyatakan dalam kata kerja diawal frase nama *use case*.
Aktor / *actor* 	Orang, proses atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu orang, biasanya dinyatakan menggunakan kata benda.
Asosiasi / *association*	Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor
Ekstensi / *extend* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walaupu tanpa *use case* tambahan itu, iasanya *use case* tambahan memiliki nama dengan yang sama dengan *use case* yang ditambhkan.
Generalisasi / *generalization* 	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case.*

Tabel 2.1. (lanjutan).

Simbol	Deskripsi
Menggunakan / *include* / *uses* 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankannya *use case* ini.

Sumber : Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. Rosa A. S. dan M. Shalahudin

b. Activity diagram

Menurut Rosa dan Shalahuddin (2016) dalam bukunya yang berjudul *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*, diagram aktivitas atau *activity diagram* menggambarkan *workflow* (alran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktiitas menggambarkan aktivitas yang dapat dlakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

1) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisakn
2) Urutan atau pengelompokan tampilan dari sistem / *user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan
3) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujinya
4) Rancangan menu yang ditampilkan pada perangkat lunak

Berikut adalah simbol-simbol yang ada pada diagram aktivitas:

Tabel 2.2. Simbol *Activity diagram*.

Simbol	Deskripsi
Status awal 	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal
Aktivitas 	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja
Percabangan / *decision* 	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu
Penggabungan / *join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu
Status akhir 	Satus akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah atatus akhir
Swimlane  Atau 	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi

Sumber : Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. Rosa A. S. dan M. Shalahudin

c. Class diagram

Menurut Rosa dan Shalahuddin (2016) dalam bukunya yang berjudul *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*, diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1) Atribut merupakan variabel-variabel yang dimiliki oleh sesuatu kelas.
2) Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Banyak berbagai kasus, perancangan kelas yang dibuat tidak sesui dengan kelas-kelas yang dibuat pada perangkay lunak, sehingga tidaklah ada gunanya lagi sebuah perancangan karena apa yang dirancang dan hasil jadinya tidak sesuai.

Kelas-kelas yang ada pada struktur sistem harus dapat melkukan fungsi-fungsi sesuai dengan kebutuhan sistem sehingga pembuat perangkat lunak atau *programmer* dapat membuat kelas-kelas di dalam progrm perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

1) Kelas main

   Kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
2) Kelas yang menangani tampilan sistem (*view*)

   Kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
3) Kelas yang diambil dari pendefinisian *use case* (*controller*)

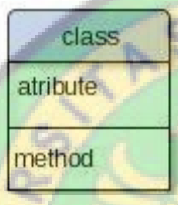
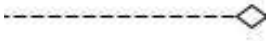
   Kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*, kelas ini biasanya disebut dengan kelas proses yang menangan proses bisnis pada perangkat lunak.
4) Kelas yang diambil dari pendefinisian data (*model*)

   Kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Semua tabel yang dibuat di basis data dapat dijadikan kelas, namun untuk tabel dari hasil relasi atau atribut mutivalue pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggungjawabkan atau tetap di dalam perancangan kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas:

Tabel 2.3. Simbol *class diagram*.

Simbol	Deskripsi
Kelas 	Kelas pada struktur sistem
Asosiasi	Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah 	Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain.
generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum khusus)
Ketergantungan 	Relasi antarkelas dengan makna kebergantungan antarkelas.
Agregasi 	Relasi antarkelas dengan makna semua-bagian (*whole-part*).

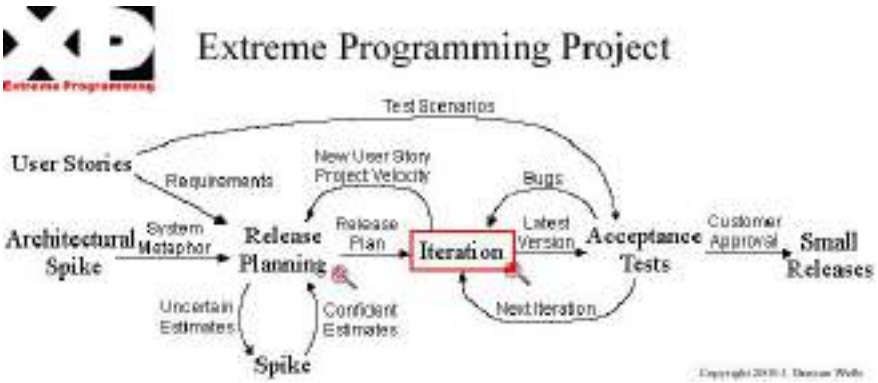
Sumber : Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek. Rosa A. S. dan M. Shalahudin

2.7. EXTREME PROGRAMMING

Extreme programming (XP) merupakan salah satu cabang dari metode *Agile* sebagai pengembangan perangkat lunak yang digunakan untuk menyesuaikan kebutuhan pengembangan. XP digunakan dalam merancang bangun aplikasi untuk memprediksi kelulusan (Rusdiana & Marfuah, 2017), dengan hasil bahwa metode XP dapat diterapkan dengan waktu pembangunan aplikasi yang tidak lama dan sesuai dengan penggunaan pengembangan perangkat lunak. XP juga digunakan sebagai pengembangan E-Keuangan dan hasil dari penelitian, yaitu aplikasi yang dibangun sederhana tanpa mengurangi kualitas dari aplikasi tersebut dengan menerapkan metode XP (Oktaviani & Hutrianto, 2016).

Penelitian lain yang menggunakan *Extreme Programming* dilakukan pada perancangan sistem informasi manajemen terpadu (SIMANTEP) dengan tahap pengembangan yang meliputi eksplorasi, perencanaan, iterasi pengembangan sistem, produksi, pemeliharaan dan diakhiri dengan publikasi sistem. Hasil penelitian ini menyimpulkan bahwa penerapan *Extreme Programming* sangat sesuai karena komunikasi antara pengembang dan client sangat berperan dalam proses perancangan perangkat lunak dan metode ini sesuai dengan *client* yang belum sepenuhnya paham atas kebutuhan dasar dari sistem yang diinginkan (B.A. Candra, K. Muludi dan A.R. Irawati. 2012).

Pengembangan aplikasi pada penelitian ini menggunakan metode *Extreme Programming* (XP). Metode XP memiliki 4 nilai dasar yang menjadi inti pokok metode XP, yaitu *Communication* (Komunikasi), *Simplicity* (Kesederhanaan), *Feedback* (Umpan Balik), dan *Courage* (Keberanian). Keempat nilai dasar ini menunjukan bahwa XP bersifat fleksibel terhadap perubahan-perubahan yang diminta oleh klien.



Gambar 2.1. Aturan pada *extreme programming*.

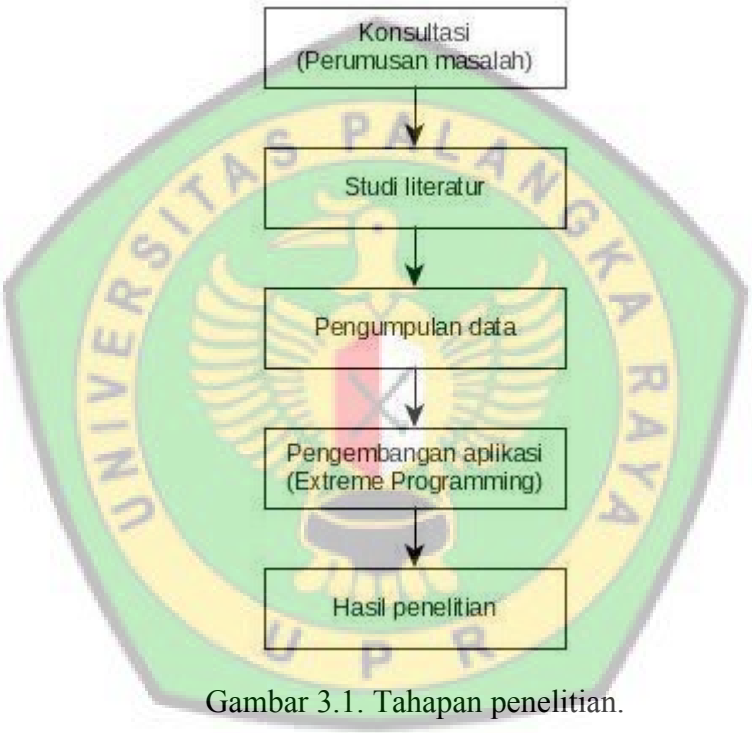
Sumber : extremeprogramming.org

Model ini cenderung menggunakan pendekatan *Object-Oriented*. Tahapan-tahapan yang harus dilalui antara lain *architectural spike*, *planning*, iterasi pengembangan (*design*, *coding*, dan *testing*). Sasaran *Extreme Programming* adalah tim yang dibentuk berukuran antara kecil sampai medium saja, tidak perlu menggunakan sebuah tim yang besar. Hal ini dimaksudkan untuk menghadapi *requirements* yang tidak jelas maupun terjadinya perubahan-perubahan *requirements* yang sangat cepat.

BAB III
METODOLOGI PENELITIAN

3.1. TAHAP-TAHAP PENELITIAN

Tahapan dari penelitian yang dilakukan meliputi konsultasi, studi literatur, pengumpulan data, dan mengembangkan perangkat lunak sesuai dengan metode pengembangan perangkat lunak yang digunakan. Berikut ini adalah model dari penelitian yang dilakukan.

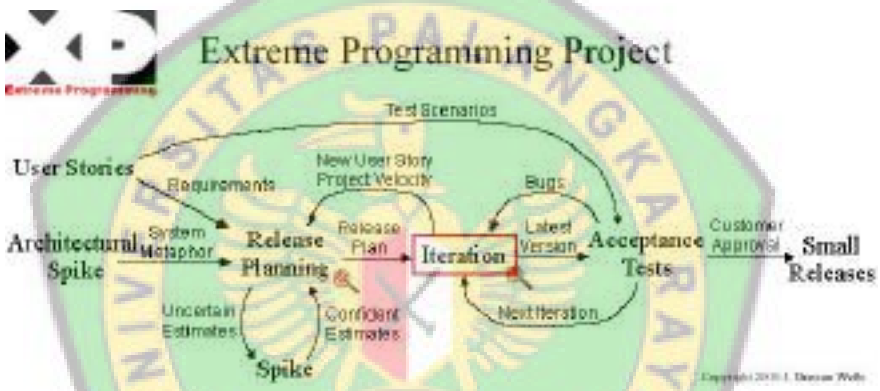


Gambar 3.1. Tahapan penelitian.

Dalam tahap konsultasi dengan dosen pembimbing, dilakukan perumusan masalah untuk menentukan masalah apa yang akan diselesaikan melalui penelitian. Didapatkan masalah yang akan diselesaikan, yaitu permasalahan informasi penutupan jalan. Dalam perumusan masalah ini juga dirumuskan mengenai *scope* penelitian dan solusi yang bisa ditawarkan dari pemelitian. Solusi yang ditawarkan adalah membuat aplikasi informasi penutupan jalan dengan menerapkan *Location Based Service* dan struktur data Graf. Studi literatur

dilakukan dengan mengumpulkan penelitian-penelitian yang berkaitan dengan permasalahan ini. Teori-teori dan hasil penelitian dari literatur tersebut digunakan sebagai bahan acuan dalam proses pengembangan aplikasi. Dari studi literatur itu pula akan diketahui data apa saja yang diperlukan.

Dilakukan pengumpulan data sesuai dengan kebutuhan sistem. Setelah semua data yang diperlukan sudah tersedia, maka akan dilakukan proses pengembangan aplikasi dengan mengadopsi salah satu metode pengembangan perangkat lunak, yaitu *Extreme Programming* atau XP. Berikut ini adalah aturan pengembangan dengan *Extreme Programming* yang didapatkan dari *website* resmi *Extreme Programming*.



Gambar 3.2. Aturan pada extreme programming.
Sumber : extremeprogramming.org

Gambar 3.2 menjelaskan siklus yang dilakukan selama proses pengembangan perangkat lunak berlangsung. Metode XP tidak terpaku pada sebuah dokumentasi awal. Dengan metode ini, dimungkinkan adanya penambahan *stories*/kebutuhan *user* yang menyebabkan adanya penambahan atau perubahan fitur pada aplikasi yang dibangun. Sehingga pengembangan aplikasi terbagi menjadi beberapa terasi pengembangan. Banyaknya iterasi pengembangan tergantung dengan *testing* yang dilakukan. Jika setelah hasil dari *testing* sudah memenuhi *user stories* dan tidak ada penambahan kebutuhan *user*, maka akan dilakukan *small release* untuk melakukan testing secara keseluruhan oleh *user* dan merangkum setiap progress yang didapatkan dari setiap iterasi pengembangan sistem. Secara garis besar berikut ini penjelasan siklus dari metodologi XP.

1) Architectural Spike

Pengembangan aplikasi dimulai dengan eksplorasi kebutuhan *user* dengan membuat *user stories*. *user* stories berisi kebutuhan *user* yang harus terpenuhi pada sistem nantinya. Sehingga dari *user stories* akan menghasilkan gambaran fungsionalitas sistem.

2) Release Planning

Dalam perencanaan pengembangan ini akan dibuat pembagian iterasi pengembangan serta skenario pengujiannya berdasarkan *user stories*. Sehingga developer yang dalam hal ini adalah peneliti mengetahui apa yang harus dicapai setelah dilakukan pembuatan program dalam setiap iterasi.

3) Iteration

Pada metodologi XP ini dimungkinkan adanya penambahan kebutuhan *user* ditengah-tengah pengembangan aplikasi. Sehingga pengembangan sistem dibuat menjadi lebih dari 1 iterasi. Pada setiap iterasinya akan dilakukan implementasi program, yang kemudian program akan diuji untuk didapatkan hasil dan mengevaluasi apakah sesuai dengan *planning* yang telah dibuat.

4) Acceptance Test

Pengujian unit dilakukan terhadap fungsi program yang telah dibuat. Pengujian dilakukan berdarakan skenario yang telah dibuat sebelumnya. Hasil dari pengujian ini akan dievaluasi yang selanjutnya akan diputuskan apakah akan dilanjutkan ke *small release* atau ada penambahan fitur pada *user stories* atau bahkan ada *bugs*/*error* yang didapatkan saat pengujian.

5) Small Release

Tahap *small release* adalah tahap dimana sistem sudah siap untuk diuji secara keseluruhan. Sehingga pada tahap ini, akan didapatkan *progress* dari setiap iterasi yang telah dilakukan sebelumnya dari awal hingga akhir pengembangan.

3.2. LOKASI DAN DATA PENELITIAN

Penentuan lokasi penelitian untuk membatasi ruang lingkup penelitian. Lokasi penelitian akan berhubungan langsung dengan data yang akan digunakan dan diolah selama penelitian.

3.2.1. Lokasi Penelitian

Lokasi yang digunakan dalam penelitian ini adalah sebagian wilayah kecamatan Jekan Raya kota Palangka Raya. Wilayahnya tersebar di dalam ruang lingkup dengan titik terluarnya adalah sebagai berikut.

5) Bundaran Besar
6) Persimpangan Jl. Yos Sudarso / Jl. Bukit Kaminting
7) Persimpangan Jl. Bukit Kaminting / Jl. Garuda
8) Persimpangan Jl. Garuda / Jl. Tjilik Riwut

Alasan pemilihan lokasi ini antara lain karena penduduknya cukup banyak, mencakup kawasan kampus Universitas Palangka Raya dan lalu lintasnya cukup padat. *Visual* dari lokasi penelitian terlampir.

3.2.2. Metode Pengumpulan Data

Untuk mendapatkan data yang dibutuhkan dalam penelitian, maka dilakukan teknik pengumpulan data berupa teknik observasi. Teknik observasi dilakukan untuk mengumpulkan data lokasi persimpangan jalan. Observasi yang dilakukan adalah dengan memanfaatkan aplikasi *Google maps* untuk mendapatkan data lokasi persimpangan jalan berupa nilai latitude dan longitude. Data ini digunakan untuk membuat representasi jalan dengan menggunakan teori Graf. Selain data lokasi persimpangan, juga diperlukan beberapa data lokasi yang digunakan untuk pengujian simulasi penutupan jalan dan rute alternatif. Data yang didapatkan terlampir.

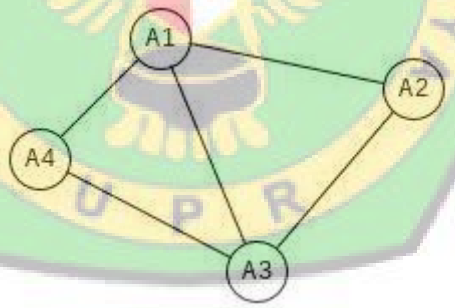
3.2.3. Metode Pengolahan Data

Terdapat 2 topik utama pada penelitian ini, yaitu *Location Based Service* (LBS) dan Graf. Berkaitan dengan LBS, pada penelitian ini akan dilakukan

pemanfaatan data koordinat lokasi sebuah perangkat untuk diolah menjadi informasi penutupan jalan. Selain itu, data-data lokasi tersebut juga diolah menjadi representasi jalan dengan menggunakan teori Graf. Setiap *node* pada Graf merepresentasikan satu data lokasi persimpangan dengan tiap-tiap *node* memiliki informasi *node* tetangga. Dengan begitu setiap *node* akan terhubung dengan *node* lainnya dan terbentuk serangkaian *edge* (sisi) yang merepresentasikan sebuah jalan. Representasi jalan ini digunakan untuk pembuatan salah satu fungsi yang dibuat pada aplikasi, yaitu pencarian rute alternatif. Berikut ini adalah *sample* representasi Graf menggunakan model *Adjacency List*.

Tabel 3.1. Adjacency List.

Node	***Node* tetangga**
A1	A2, A3, A4
A2	A1, A3
A3	A1, A2, A4
A4	A1, A3



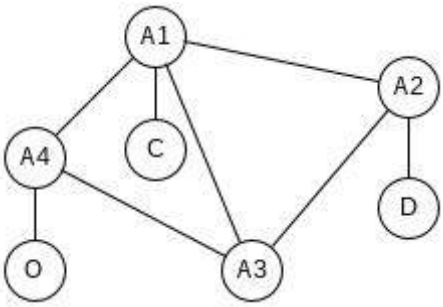
Gambar 3.3. Contoh representasi jalan menggunakan Graf.

Dalam penggunaannya, akan ada *node* baru berupa titik penutupan jalan, titik asal dan titik tujuan. Agar *node-node* tersebut dapat terhubung dengan Graf yang sudah ada sebelumnya, maka dibuatlah sebuah *rules* untuk menghubungkan *node* baru tersebut yang nantinya akan diolah untuk kebutuhan dalam menentukan rute alternatif. Berikut ini *rules*-nya.

Gambar 3.4. *Flowchart rules* untuk menghubungkan *node* baru.

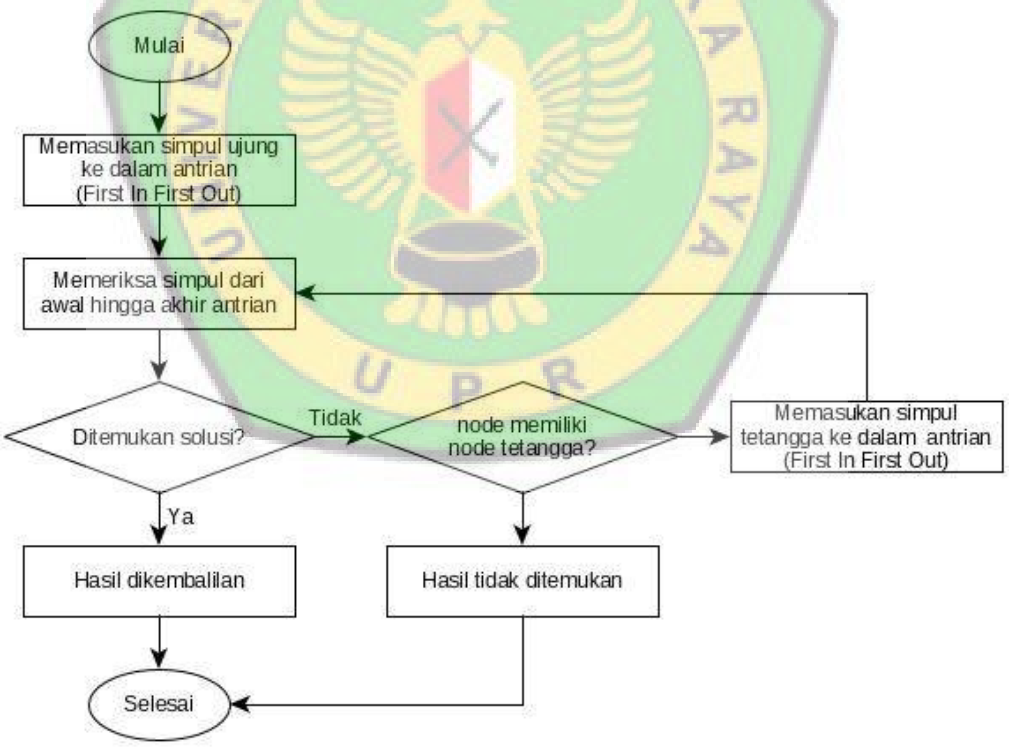
Ketiga jenis data baru disimpan dalam variabel sebagai *node* dengan label c (*closed road*/titik penutupan), o (*origin*/titik asal) dan d (*destination*/titik tujuan). Dilakukan perhitungan jarak antara masing-masing *node* baru dengan setiap *node* yang sudah ada. Dari perhitungan tersebut, didapatkan 1 *node* dengan jarak terdekat dengan masing-masing *node* baru. *Node* terdekat itulah yang akan diatur dan ditetapkan sebagai *node* tetangga dari masing-masing *node* baru tersebut.

Hasil dari *rules* ini adalah data baru dapat teraplikasikan sebagai *node* pada graf. *Node* ini bersifat sementara hanya untuk kebutuhan mendapatkan rute alternatif menggunakan algoritma *Breadth First Search*. Berikut ini contoh hasil penambahan *node* baru pada graf.



Gambar 3.5. Contoh Graf setelah penambahan *Node* baru.

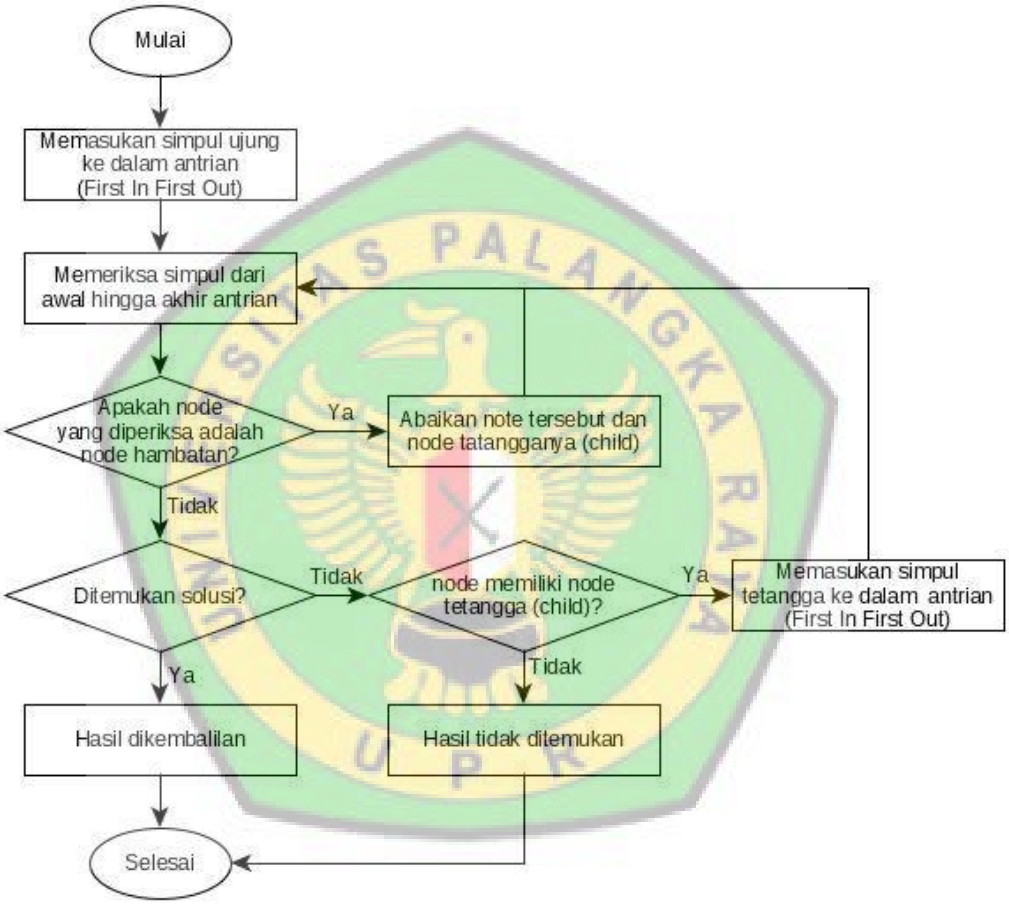
Seperti yang telah dijelaskan sebelumnya, untuk mendapatkan rute alternatif menerapkan algoritma *Breadth First Search*. Rute yang dihasilkan bukan merupakan rute terpendek/optimal, melainkan sebatas rute alternatif yang menghindari titik penutupan jalan. Berikut ini representasi algoritma BFS dalam bentuk *flowchart*.



Gambar 3.6. *Flowchart* algoritma BFS.

Untuk sistem yang akan dibuat, terdapat penambahan proses dalam algoritma tersebut. Proses tersebut adalah proses pemeriksaan *node* atau titik hambatan

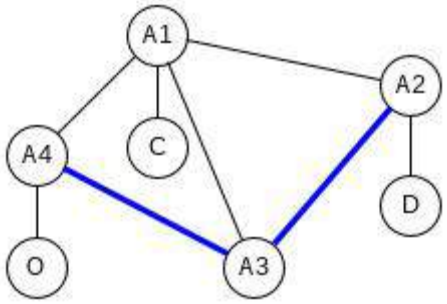
(penutupan). Ketika saat melakukan penelusuran menemukan *node* hambatan, maka *node* tetangga atau *child* dari *node* tersebut akan dilewati atau diabaikan dalam pencarian solusi. Kemudian pemeriksaan dilanjutkan ke *node* yang tersisa di antrian. Sehingga tidak terdapat titik hambatan pada rute yang dihasilkan, sesuai dengan tujuan dari fungsi yang dibuat yaitu mencari rute alternatif. Berikut ini algoritmanya setelah dilakukan penambahan.



Gambar 3.7. *Flowchart* algoritma BFS setelah penambahan proses.

Terlihat bahwa *node* yang ada pada antrian tidak serta merta diperiksa sebagai solusi atau bukan, melainkan akan dilakukan pemeriksaat apakah *node* tersebut meruapakan *node* hambatan atau bukan. Nantinya sistem akan dibuat memungkinkan adanya lebih dari 1 *node* hambatan. Sehingga pemeriksaan *node* hambatan dilakukan sebanyak penutupan jalan yang terdata. Berikut ini adalah

contoh hasil simulasi yang diharapkan dari *request* rute yang mengembalikan rute alternatif.



Gambar 3.8. Contoh simulasi rute alternatif.

3.3. EKSPLORASI KEBUTUHAN USER

Dilakukan analisis terhadap sistem yang akan dibuat dengan melakukan eksplorasi terhadap kebutuhan *user*. Pada tahap eksplorasi ini, kebutuhan *user* dijabarkan dengan menggunakan *user Stories*. *user Stories* dideskripsikan pada tabel 3.2 berikut ini.

Tabel 3.2. *user Stories*.

Judul	Deskripsi	Acceptance Criteria
Melakukan registrasi dan login pada aplikasi *mobile*.	Sebagai *user public*, saya ingin melakukan registrasi untuk memiliki akses pada aplikasi.	Terdapat *form* registrasi dan login untuk memiliki akses ke aplikasi *mobile*.
Membuat informasi penutupan jalan	Sebagai *user public*, saya ingin memiliki akses untuk dapat membuat informasi.	Terdapat *form* untuk membuat informasi penutupan jalan pada aplikasi *mobile*
Melihat lokasi penutupan pada *map*.	Sebagai *user public*, saya ingin melihat lokasi penutupan jalan pada tampilan *map*.	Terdapat tampilan *map* yang menampilkan titik penutupan jalan.

Tabel 3.2. (lanjutan).

Judul	Deskripsi	Acceptance Criteria
Melakukan *request* rute alternatif.	Sebagai *user public*, saya ingin melakukan *request* rute alternatif untuk menghindari penutupan jalan.	Dapat melakukan *request* rute dengan memasukan titik asal dan tujuan, kemudian ditampilkan rute.
Membagikan informasi via aplikasi eksternal.	Sebagai *user public*, saya ingin membagikan informasi penutupan jalan menggunakan aplikasi *messenger*.	Terdapat opsi membagikan informasi via aplikasi eksternal pada perangkat *mobile* pengguna.
Melakukan registrasi dan login pada aplikasi *website*.	Sebagai *operator*, saya ingin melakukan registrasi untuk memiliki akses pada aplikasi.	Terdapat *form* registrasi dan login untuk memiliki akses ke aplikasi *mobile*.
Mengelola informasi penutupan jalan.	Sebagai *operator*, saya ingin mengelola informasi dengan *approve* dan hapus informasi.	Terdapat halaman yang menyajikan informasi penutupan jalan, yang dilengkapi dengan tombol *approve* dan hapus.
Melihat data *user public*.	Sebagai *operator*, saya ingin melihat data *user public* yang membuat informasi penutupan jalan.	Terdapat halaman yang menampilkan data *user public* yang telah melakukan registrasi.

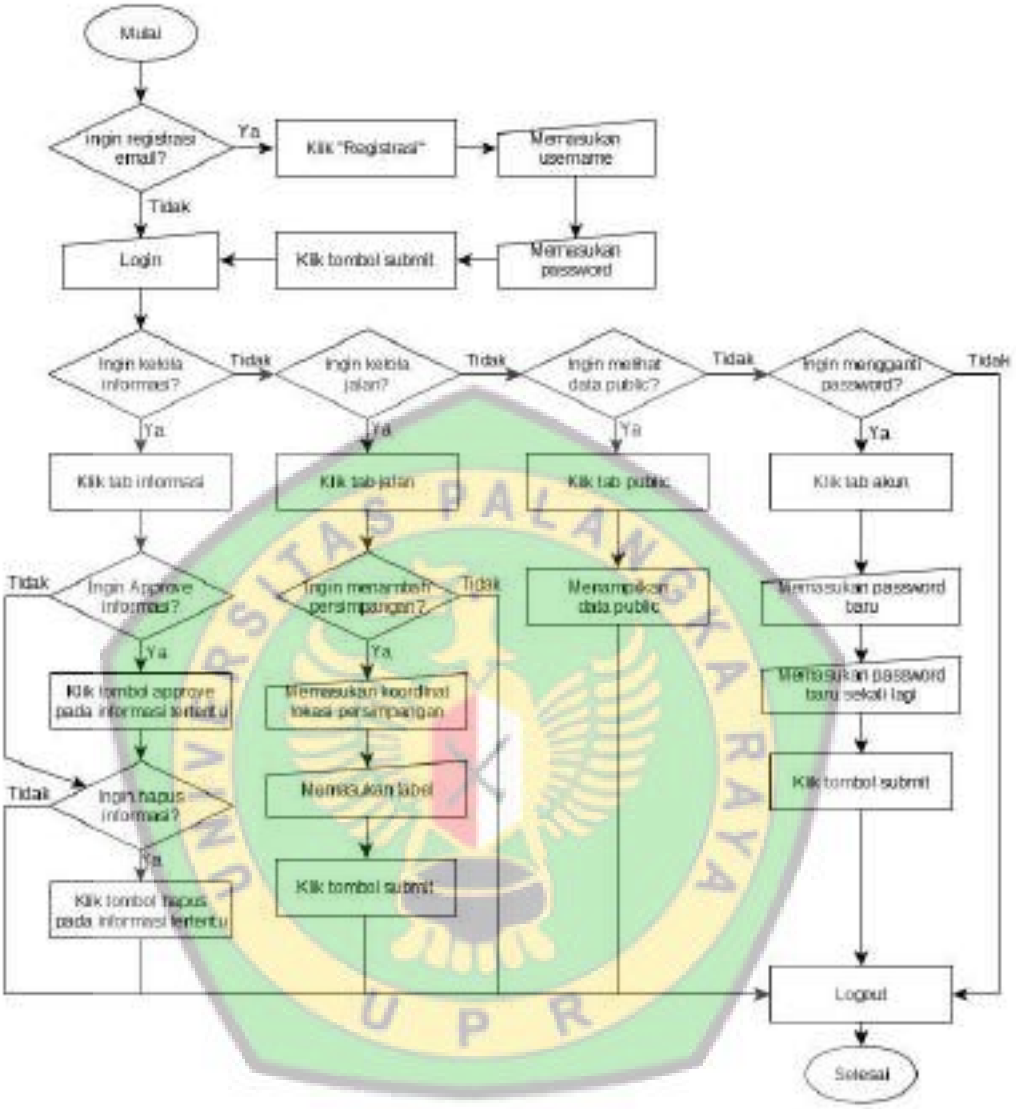
Tabel 3.2. (lanjutan).

Judul	Deskripsi	Acceptance Criteria
Menambah titik jalan.	Sebagai *operator*, saya ingin menambah titik jalan agar rute yang dihasilkan mencakup wilayah yang lebih luas.	Terdapat *form* untuk menambah titik jalan.
Mengganti *password* Akun *operator*.	Sebagai *operator*, saya ingin mengganti *password* akun saya.	Terdapat *form* untuk mengganti *password* akun *operator*.

Dari *user stories* tersebut, didapatkan semua kebutuhan *user*. Dengan mengacu *user stories* tersebut, dilakukan analisis terhadap kebutuhan sistem. Untuk mengetahui kebutuhan sistem, maka dibuatlah *flowchart* sistem untuk menggambarkan alur kerja sistem yang akan dibuat.

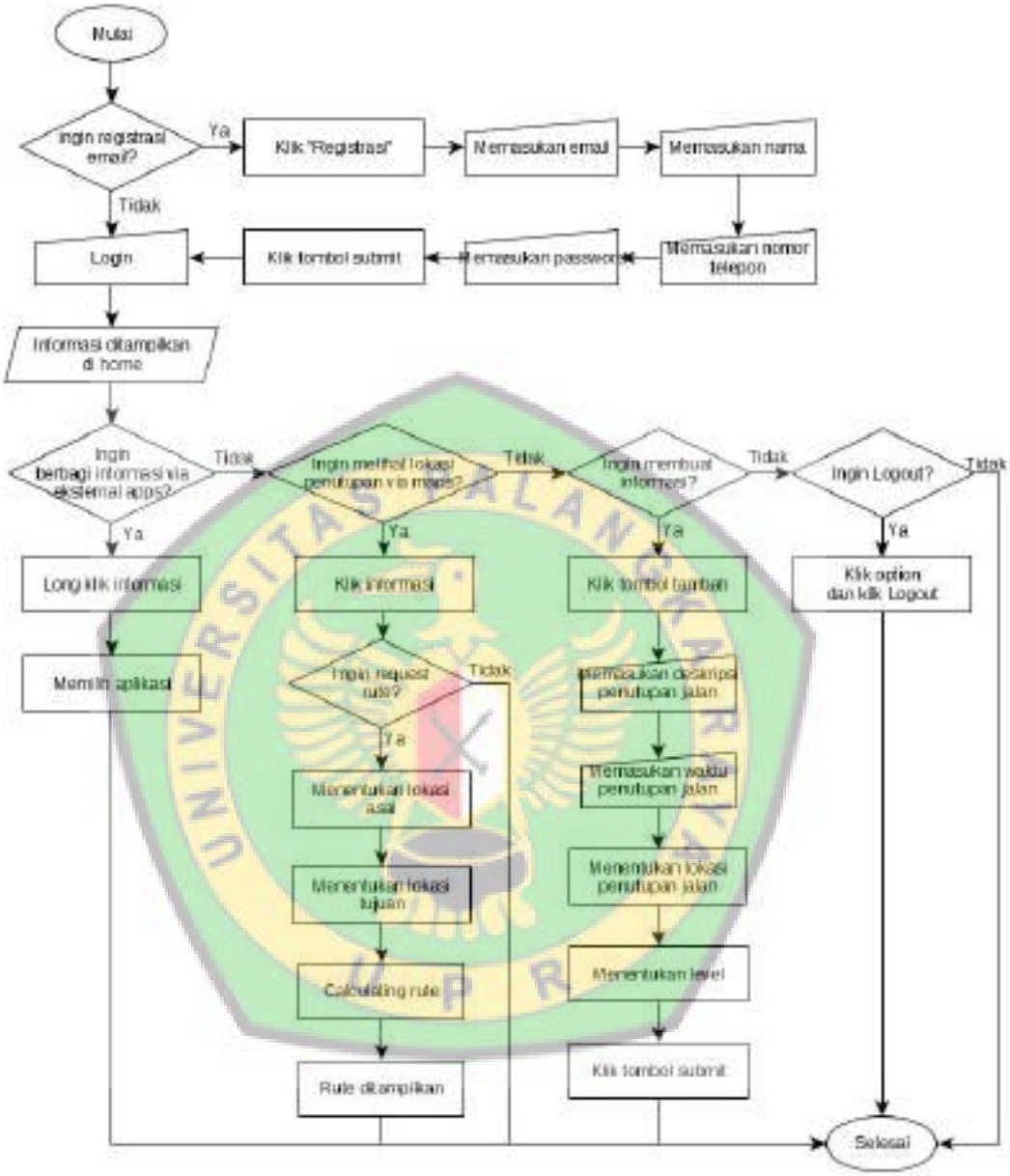
Flowchart sistem terbagi menjadi 2, yaitu *flowchart* untuk sisi *public* dan *flowchart* untuk sisi *operator*. Berikut ini *flowchart* yang disajikan pada gambar 3.10 dan 3.11.

Operator



Gambar 3.9. *Flowchart* sistem untuk sisi *operator*.

Public



Gambar 3.10. *Flowchart* sistem untuk sisi *public*.

Flowchart pada gambar 3.9 dan 3.10 menggambarkan alur kerja sistem secara keseluruhan dari sisi *public* dan *operator*. Dari kedua *flowchart* tersebut, diketahui fungsionalitas sistem yang harus dipenuhi adalah sebagai berikut.

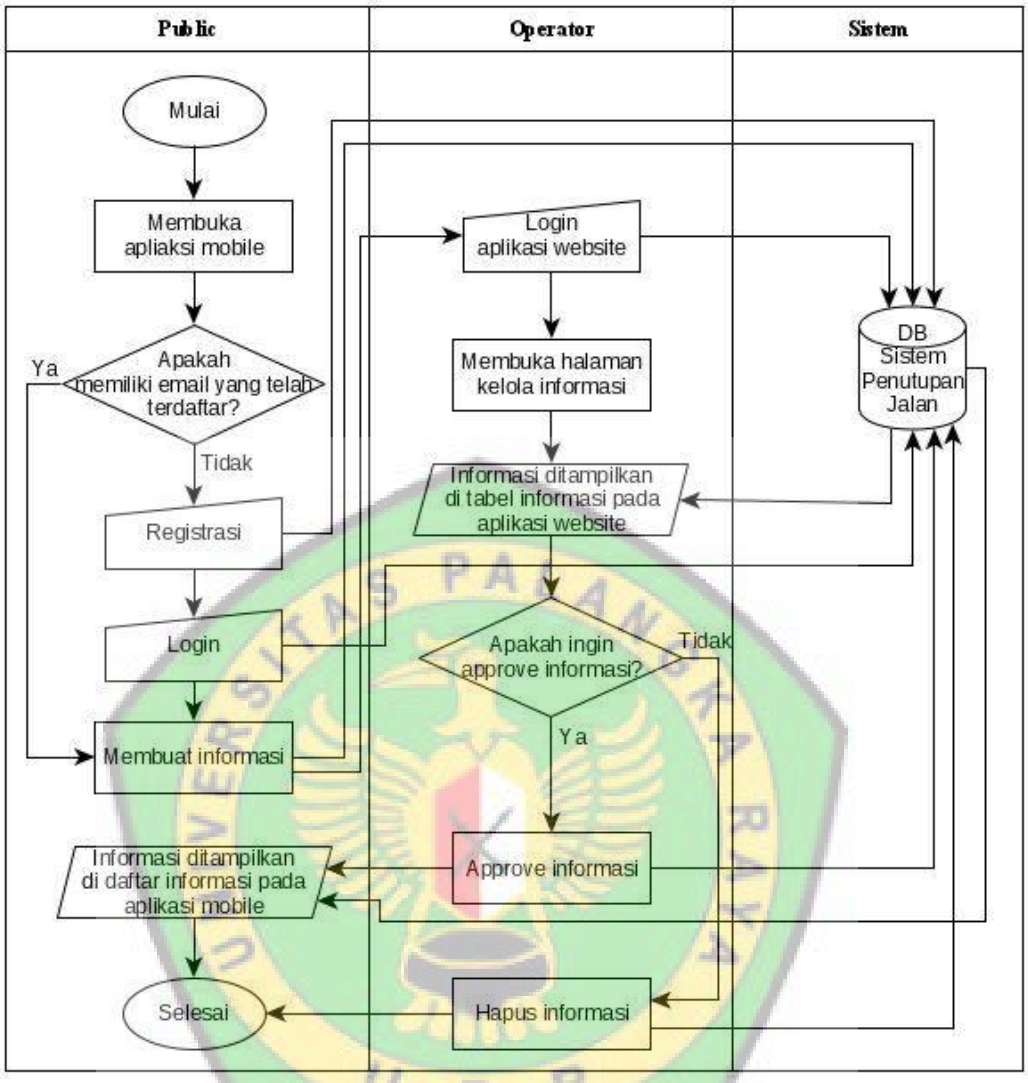
a. Operator

1) Registrasi dan *Login*.
2) Kelola informasi, berupa approve dan hapus informasi.
3) Menambahkan koordinat baru.
4) Melihat data *user public*.
5) Mengganti *password*.

b. Public

1) Registrasi dan *Login*.
2) Lihat informasi.
3) Buat informasi.
4) Berbagi informasi via aplikasi eksternal.
5) *request* rute alternatif.

Adapun rancangan mekanisme pembuatan informasi hingga tampil di daftar informasi pada aplikasi *mobile* adalah sebagai berikut.



Gambar 3.11. Mekanisme proses pembuatan informasi penutupan jalan.

Dari gambar 3.11, diketahui bahwa informasi penutupan jalan dibuat oleh *user public* menggunakan aplikasi *mobile*. Dalam membuat informasi baru, *user* membutuhkan email yang telah terdaftar pada sistem yang kemudian *user* bisa login dan membuat informasi. informasi yang dibuat akan disimpan ke dalam database. Agar informasi dapat ditampilkan pada daftar informasi di aplikasi *mobile*, informasi harus di-*approve* oleh *operator*. Sehingga *operator* harus masuk ke aplikasi *website* untuk mengelola informasi yang ada. *operator* akan mendapati informasi yang baru masuk. *operator* memiliki pilihan untuk *approve* atau hapus informasi. Jika informasi di-*approve*, maka informasi akan langsung

tersedia di aplikasi *mobile*. Namun jika *operator* menghapus informasi, maka informasi akan langsung dihapus dari *database*.

Untuk membangun sistem, terdapat beberapa kebutuhan nonfungsional yang digunakan selama proses pembuatan sistem. Mengacu pada batasan masalah dari penelitian ini, berikut ini kebutuhan non fungsional (penunjang) untuk pembuatan sistem.

1) Visual Studio Code
2) Android Studio
3) Browser Google Chrome
4) Postman
5) XAMPP (*localhost* untuk pengembangan sistem)
6) Web Hosting Provider
7) PC desktop
8) *mobile* device (ponsel)

3.4. ITERASI PERTAMA

Pada iterasi pertama dilakukan pengembangan sistem sesuai dengan *user stories* yang telah dibuat sebelumnya. Tahapan dalam iterasi pertama ini adalah membuat *Planning* iterasi pertama, membuat desain sistem, implementasi program dan pengujian program.

3.4.1. Planning Iterasi Pertama

Berikut ini adalah *Planning* pengembangan sistem dari iterasi pengembangan sistem pertama.

Tabel 3.3. *Planning* pengembangan sistem iterasi pertama.

Pengembangan Sistem	Acceptance Test
Registrasi dan *Login* pada aplikasi *mobile*.	*user public* dapat melakukan registrasi dan login.

Tabel 3.3. (lanjutan).

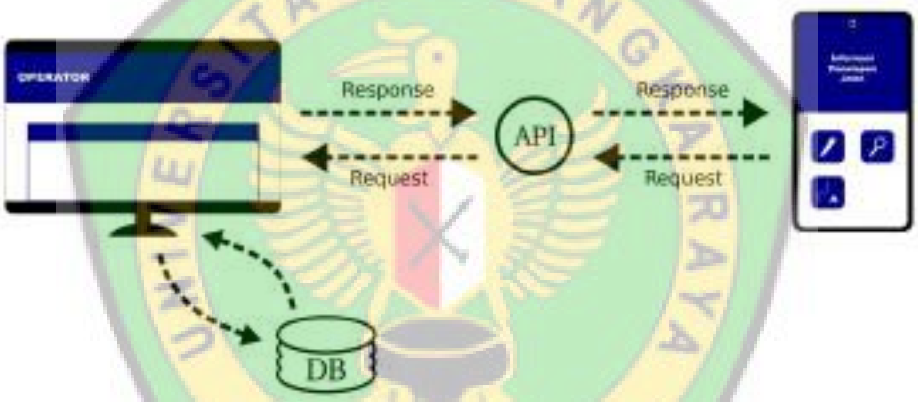
Pengembangan Sistem	Acceptance Test
Membuat informasi penutupan jalan	*user public* dapat membuat informasi penutupan jalan.
Melihat lokasi penutupan jalan pada *map*	*user public* dapat melihat titik penutupan jalan pada *map*.
Melakukan *request* rute alternatif.	*user public* dapat melakukan *request* rute alternatif.
Membagikan informasi via aplikasi eksternal.	*user public* dapat membagikan informasi via aplikasi eksternal.
Regitrasi dan Login pada aplikasi *website*.	*operator* dapat melakukan Registras dan Login.
Mengelola informasi penutupan jalan.	*operator* dapat mengelola informasi penutupan jalan, yaitu *approve* dan hapus informasi.
Melihat data *user public*.	*operator* dapat melihat data *user public*.
Menambah titik jalan.	*operator* dapat menambah titik koordinat jalan.
Mengganti *password* Akun *operator*.	*operator* dapat mengganti *password* akun masihng-masing *operator*.

3.4.2. Desain Sistem Iterasi Pertama

Desain program dibuat berdasarkan fungsionalitas yang didapatkan dari *user stories*. Selain desain program, juga terdapat arsitektur sistem yang menggambarkan komunikasi data pada sistem. Program dibangun dengan model pemrograman berorientasi objek (OOP). Desain program dibuat menggunakan *unified modelling language* (UML), yaitu *Use Case diagram*, *Activity diagram*, *Class diagram*, serta desain *user interface*.

1) Arsitektur sistem

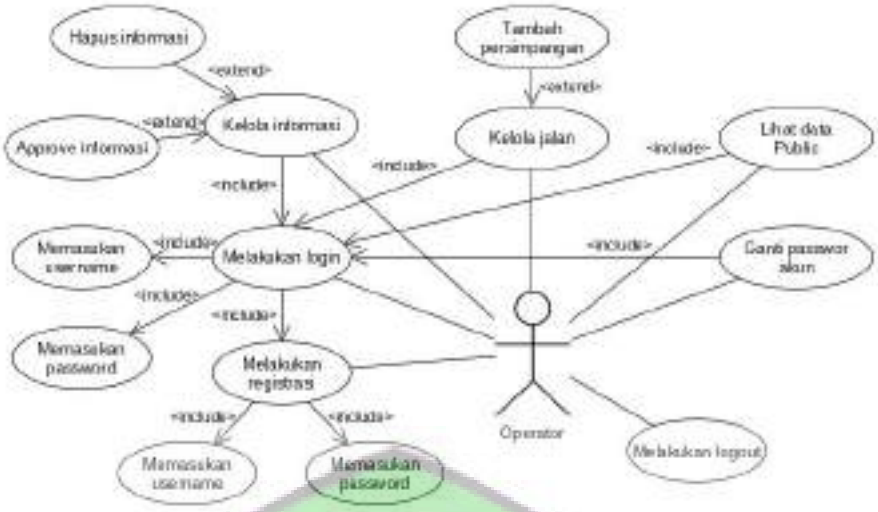
Arsitektur sistem dibuat dengan menerapkan model komunikasi *Rest Full API*. Model ini memungkinkan sebuah *database* dapat diakses oleh aplikasi lain dengan menggunakan perantara API. Pada penelitian ini, *website* yang dibuat akan berperan sebagai *Rest Server* dan aplikasi *mobile* berperan sebagai *Rest Client*. Setiap kebutuan data pada aplikasi *mobile* didapatkan dengan melakukan *request* melalui API yang disediakan *website*. Kemudian *website* mengembalikan *response* kepada aplikasi *mobile*. Komunikasi datanya menggunakan protolol *http*. Berikut ini arsitekturnya.



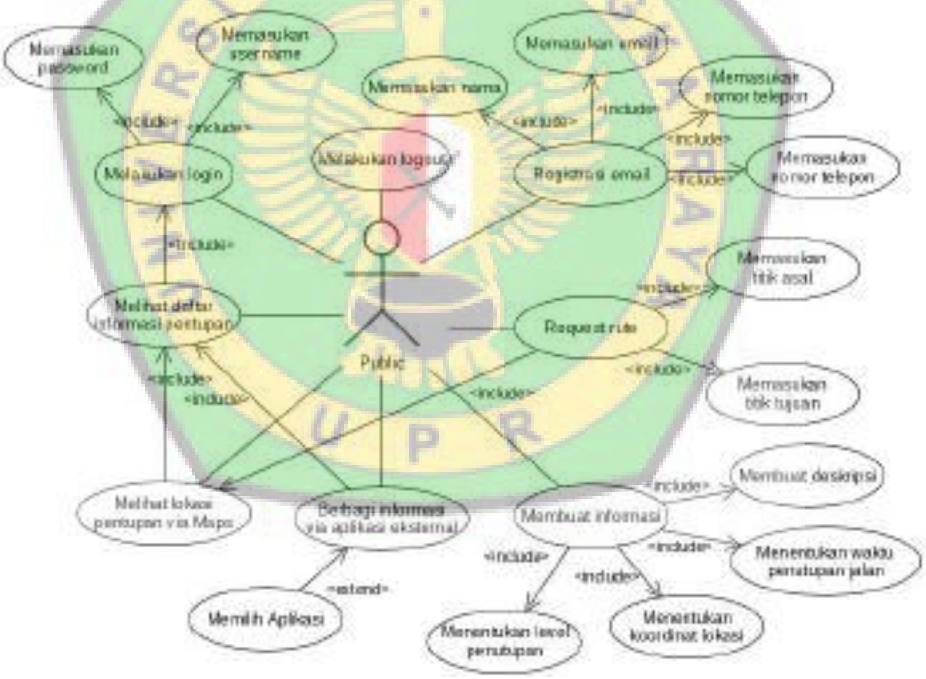
Gambar 3.12. Arsitektur Sistem.

2) Use case diagram

Use case diagram menggambarkan fungsionalitas yang terdapat dalam sistem yang dibuat. Fungsionalitas dapat dilihat dari hubungan antara *actor* (*user*) dengan kegiatan-kegiatan yang dilakukan (*case*). Berikut ini *use case* dari sistem yang dibuat.



Gambar 3.13. *Use Case operator*.



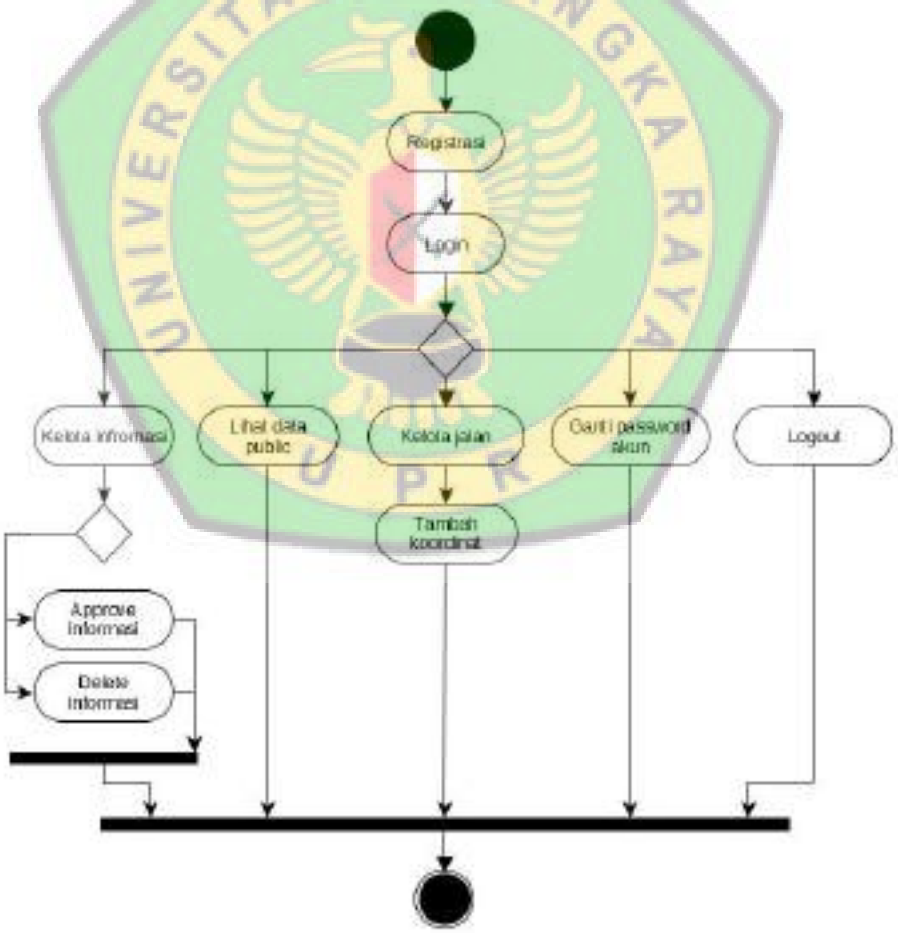
Gambar 3.14. *Use Case public*.

Dari gambar 3.13 dan 3.14, diketahui terdapat 2 aktor pada sistem, yaitu *operator* dan *public*. Beberapa kegiatan yang dilakukan *operator* diantaranya dapat melakukan registrasi dan login untuk bisa menggunakan aplikasi, selanjutnya dapat melakukan kelola informasi, kelola *public* dan kelola akun. Sedangkan

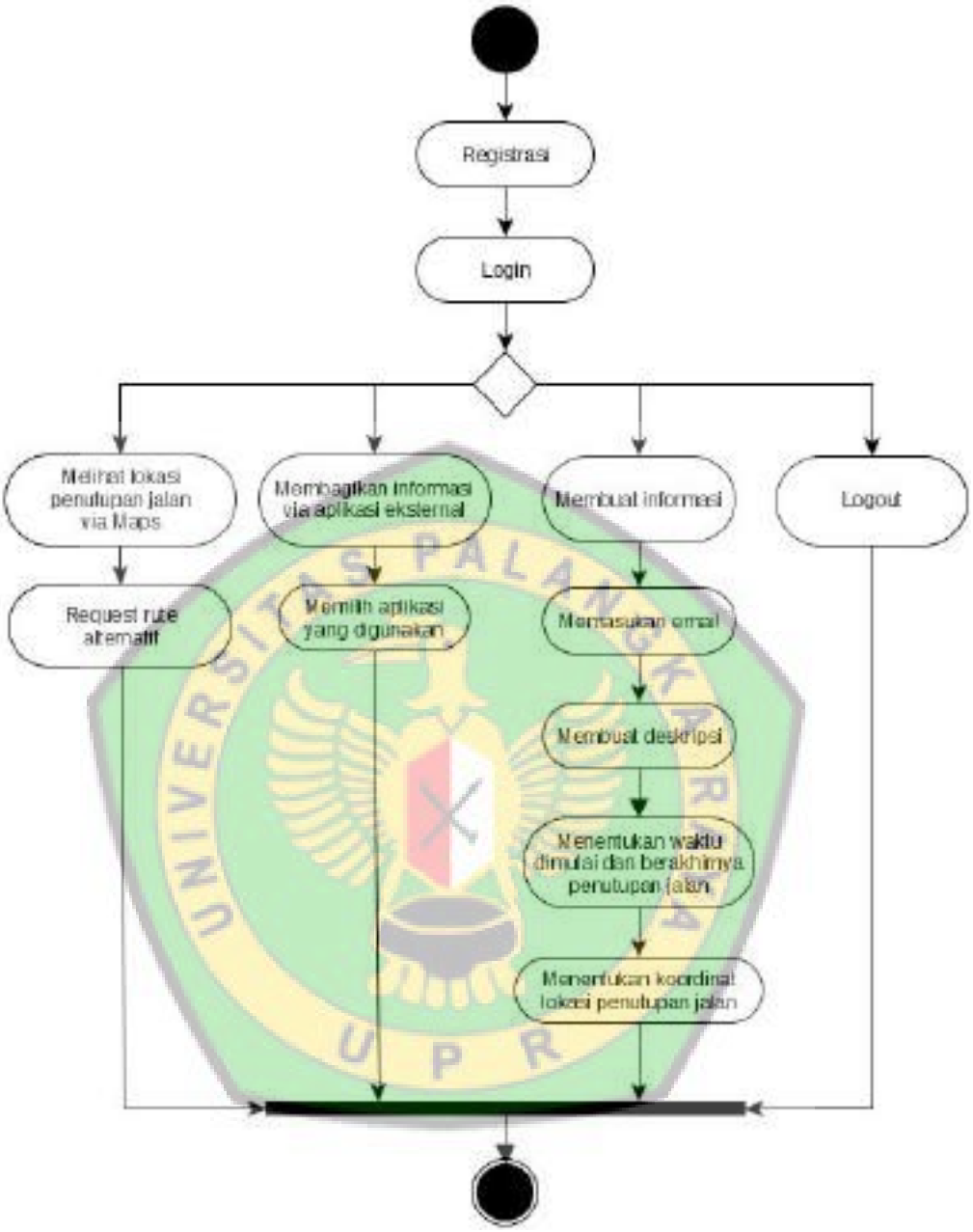
public dapat melihat informasi penutupan jalan, membagikannya via aplikasi eksternal, mendaftarkan email, melihat lokasi penutupan via *maps*, serta melakukan *request* rute alternatif. Untuk membuat informasi baru, *public* perlu melakukan registrasi terlebih dahulu dengan mendaftarkan *email*, sehingga *email* yang telah terdaftar bisa digunakan untuk membuat informasi baru.

3) Activity diagram

Diagram ini menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses dan memperlihatkan urutan aktifitas proses pada sistem. *Activity diagram* dibuat berdasarkan *use case diagram*. Terdapat 2 *activity diagram* yang digambarkan pada gambar 3.15 dan gambar 3.16 berikut ini.



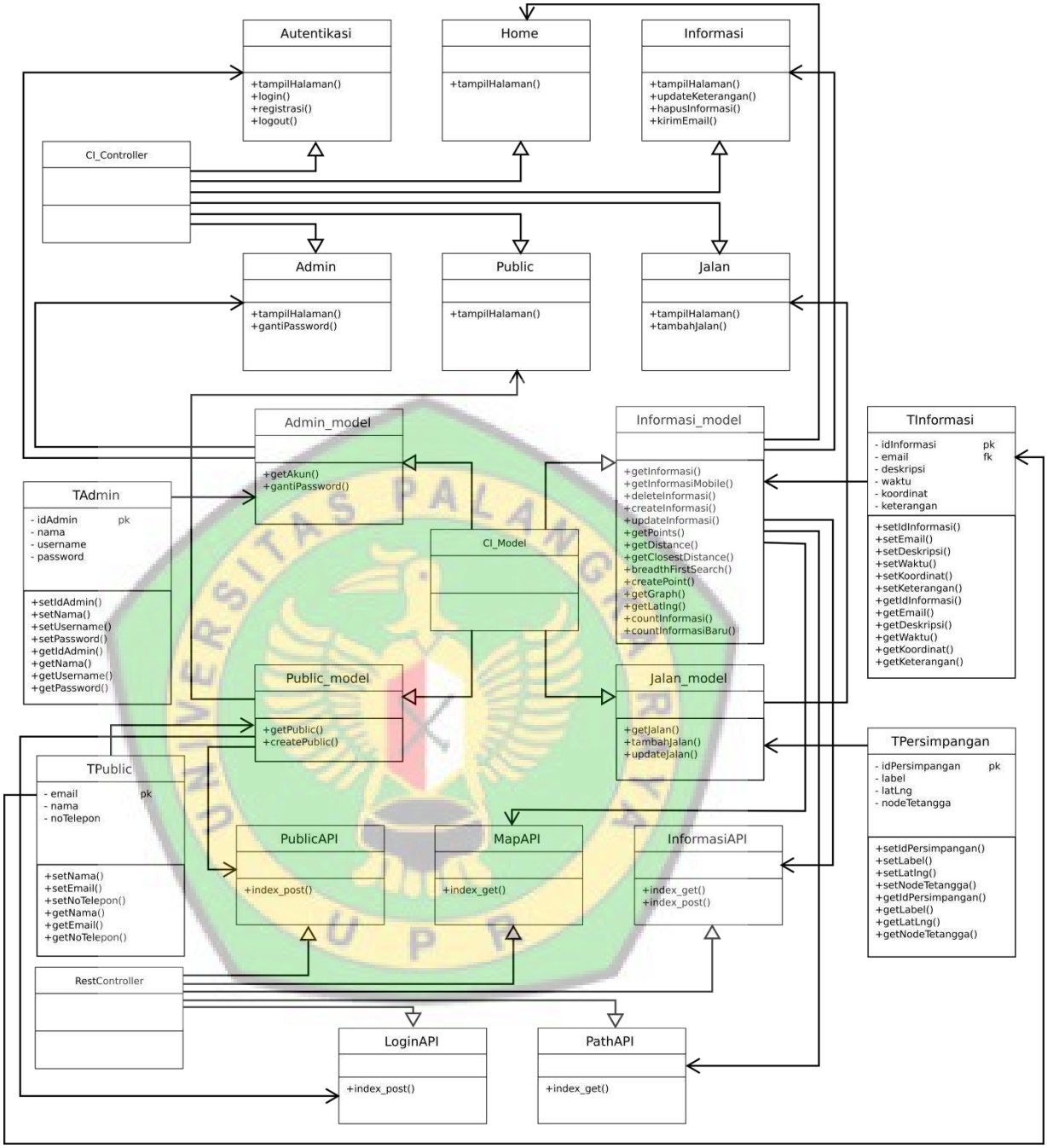
Gambar 3.15. *Activity diagram operator*.



Gambar 3.16. *Activity diagram public*.

4) Class diagram

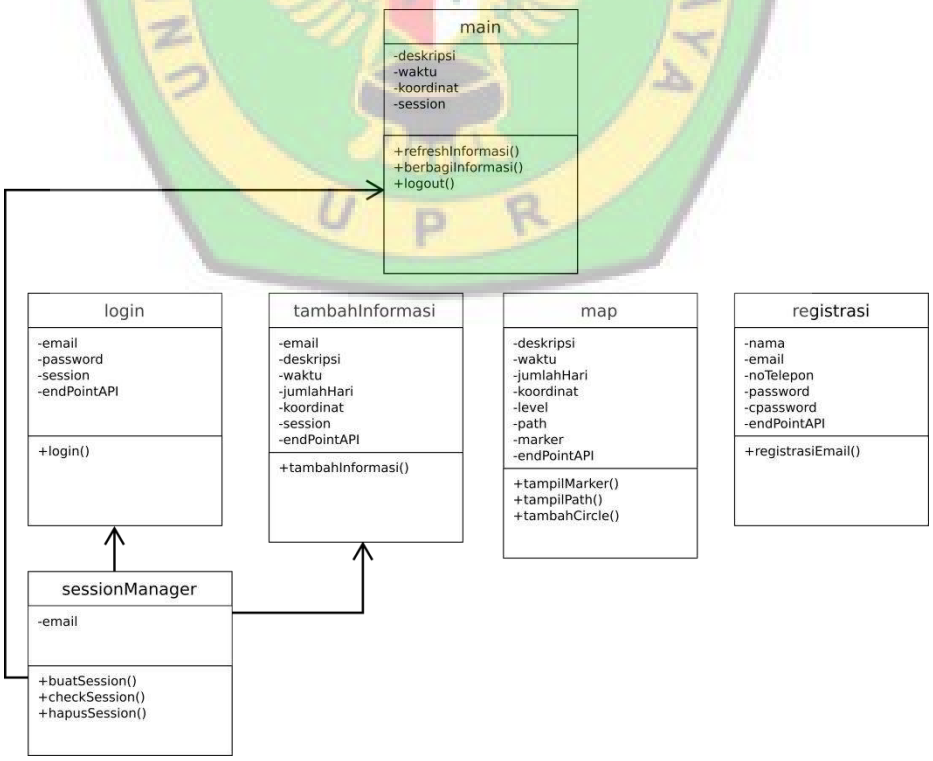
Pada sistem ini terdapat 2 buah aplikasi yang dperuntukan kepada 2 jenis aktor atau *user* yang berbeda. Untuk memberikan gambaran program dari sistem yang dibuat, dibuat 2 buah *class diagram*, yaitu *class diagram* untuk aplikasi *mobile* seperti pada gambar 3.17 dan aplikasi *website* seperti pada gambar 3.18.



Gambar 3.17. *Class diagram operator* (*Web Application*)

Class diagram pada gambar 3.17 adalah *class diagram* dari aplikasi *website* dengan arsitektur pemrograman MVC (*Model View Controller*). Program dibuat menggunakan *framework Codeigniter* dan *library Rest Full API*. Jika diperhatikan pada gambar 3.16 terdapat kelas yang atribut dan *method* nya tidak terdefinisi

pada *class diagram* tersebut, yaitu kelas *CI_Controller*, *CI_Model* dan *RestController*. Kelas *CI_Controller* dan *CI_Model* adalah *built-in class* dari C*odeigniter*. Sedangkan kelas *RestController* adalah kelas yang tersedia pada *library Rest Full API* yang digunakan untuk koneksi *http* antara aplikasi *website* dan *mobile*. Terdapat 2 jenis kelas utama, yaitu kelas *Controller* dan kelas *Model*. Kelas *Controller* berfungsi mengatur jalannya program, sedangkan kelas *Model* berfungsi untuk menjalankan *query* ke *database* untuk mendapatkan semua kebutuhan data pada aplikasi *website* maupun *mobile*. Untuk kelas API seperti *PublicAPI, MapAPI, InformasiAPI, LoginAPI,* dan *PathAPI* merupakan kelas yang akan diakses oleh kelas-kelas yang ada pada aplikasi *mobile*. Sehingga pada kelas-kelas yang ada pada aplikasi *mobile* akan mendefinisikan *end-point* berupa *url* sebagai perantara *request* yang dikirimkan dengan protokol *http* yang mengarah pada kelas API pada *server* (*website*), yang selanjutnya website akan menerima *request* dan menjalankan fungsi-fungsi tertentu sesuai dengan *request* yang dikirimkan dan dikembalikan dalam bentuk *response* kepada aplikasi *mobile*.



Gambar 3.18. *Class diagram public* (*mobile Application*)

Dari kedua gambar tersebut, diketahui bahwa aplikasi yang memiliki akses langung kepada *database* adalah aplikasi *website* yang digunakan *operator*. Sedangkan aplikasi *mobile* mendapatkan data yang diperlukan dari *database* melalui API dari *rest server* (*website operator*). Semua proses query dan logika yang berjalan seperti pencarian rute hingga menampilkan lokasi penutupan jalan terjadi pada sisi *website* yang berperan sebagai Rest Server. Sementara aplikasi *mobile* sebagai Rest Client hanya dapat mengirimkan input sebagai *request* yang dikirim ke Rest Server dan akan dikembalikan response berupa output yang diinginkan sesuai *request*.

5) Struktur Database

Struktur *database* sistem dijabarkan pada beberapa tabel dibawah ini.

Tabel 3.4. Tabel informasi

No	Nama Field	Tipe	Panjang	Keterangan
1	idinformasi	int	-	Primary key
2	email	varchar	40	Foreign key
3	deskripsi	varchar	150	-
4	waktu	varchar	30	-
5	koordinat	varchar	30	-
6	level	varchar	60	-
7	keterangan	varchar	10	-

Tabel 3.5. Tabel *public*

No	Nama Field	Tipe	Panjang	Keterangan
1	email	varchar	40	Primary key
2	nama	varchar	30	-
3	noTelepon	varchar	20	-

Tabel 3.6. Tabel *operator*

No	Nama Field	Tipe	Panjang	Keterangan
1	id*operator*	int	-	Primary key
2	*user*name	varchar	30	-
3	*password*	varchar	10	-

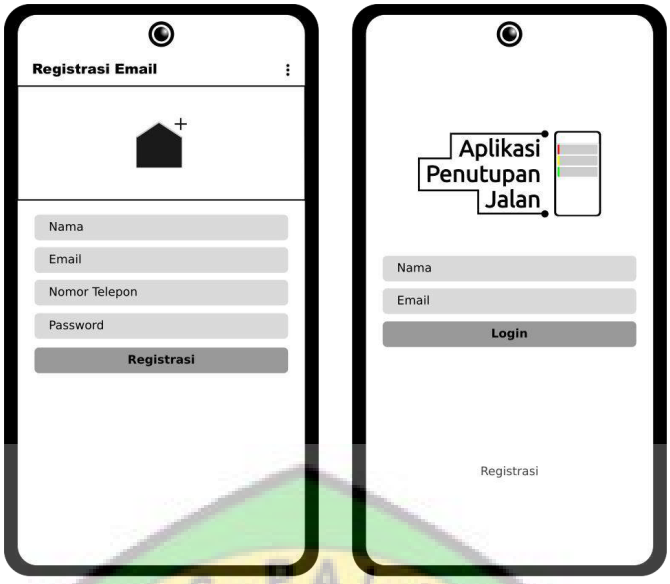
Tabel 3.7. Tabel *operator*

No	Nama Field	Tipe	Panjang	Keterangan
1	idPersimpangan	int	-	Primary key
2	label	varchar	15	-
3	koordinat	varchar	30	-
4	*Node*Tetangga	varchar	30	-

Terdapat 2 tabel yang memiliki relasi, yaitu tabel *public* dan tabel informasi. informasi yang dibuat memiliki atribut email, dimana *field* email pada tabel informasi merupakah *foreign key* yang berelasi dengan *field* email pada tabel publuc. Sehingga nantinya akan diketahui identitas *user* yang membuat informasi. Sementara tabel lainnya, yaitu tabel persimpangan berisi koordinat penyusun jalan yang diguanakan untuk mendapatkan rute alternatif, serta tabel *operator* yang berisi data *operator* atau admin.

6) *user* Interface

Berikut ini adalah rancangan atau desain *user interface* dari 2 aplikasi yang dibuat, yaitu aplikasi *website* dan aplikasi *mobile*. Desain UI dibuat menggunakan aplikasi *Inkscape*. Desain UI untuk aplikasi *mobile* disajikan pada gambar 3.19, 3.20, 3.21 dan 3.22.



Gambar 3.19. Desain UI Halaman registrasi *email* dan *login*.

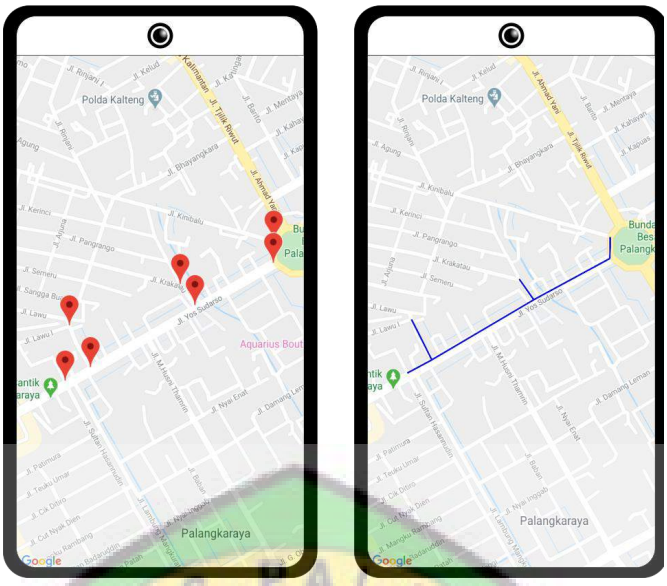
Gambar 3.19 adalah desain UI untuk halaman registrasi dan *login*. Pada halaman tersebut terdapat 4 *field*, yaitu *field* nama, email nomor telepon dan *password*. Tombol registrasit digunakan untuk *submitting* data.

Gambar 3.20. Desain UI Halaman daftar informasi, bagikan informasi dan option *logout*.

Gambar 3.20 adalah desain UI untuk Halaman utama pada aplikasi. informasi ditampilkan dalam bentuk *card view*. informasi yang ditampilkan adalah deskripsi penutupan jalan, koordinat dan waktu penutupan jalan. Pada *card view* juga terdapat bagian yang berwarna merah, kuning atau hijau yang menandakan level dari penutupan jalan. Terdapat tombol dengan tanda *plus* (+) yang mengarahkan pada halaman tambah data dan terdapat tombol 3 titik yang memberikan opsi logout.

Gambar 3.21. Desain UI halaman tambah informasi.

Gambar 3.21 adalaah desain UI untuk halaman tambah informasi. Terdapat 2 bagian utama pada halaman tambah informasi, yaitu bagian *map* dan data *entry*. Pada bagian *map*, *user* dapat menentukan lokasi penutupan jalan dengan tekan dan tahan pada daerah yang ingin dipilih. Nantinya akan muncul *marker* yang akan menandai daerah tersebut. Kemudian pada bagian data *entry*, terdapat 4 *field* yang harus diisi, yaitu *field* email, deskripsi, tanggal penutupan dan lama penutupan. Kemudian terdapat *radio button* yang berisi pilihan level dari penutupan jalan. Tombol centang putih yang berada pada *toolbar* digunakan untuk *submitting* data.



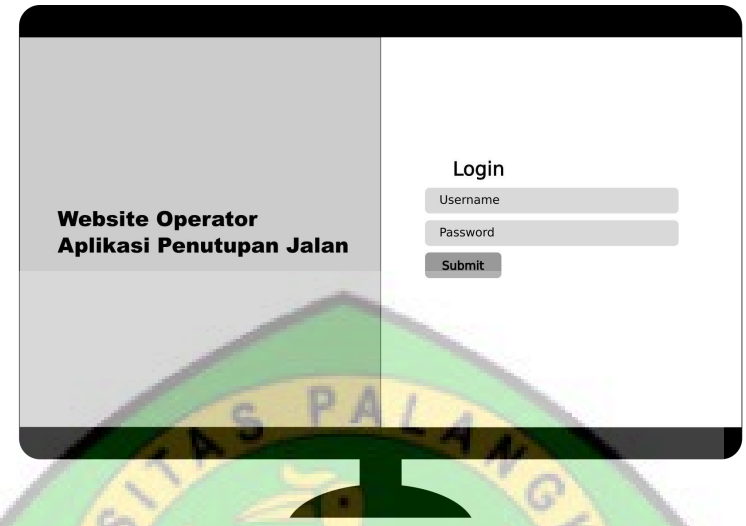
Gambar 3.22. Desain UI Halaman *maps*.

Gambar 3.22 merupakan tampilan *maps* View untuk melihat titik penutupan jalan. Pada halaman ini juga *user public* dapat melakukan *request* rute alternatif.

Desain UI untuk aplikasi *website* disajikan pada gambar 3.23, 3.24, 3.25, 3.26, 3.27, 3.28 dan 3.29.

Gambar 3.23. Desain UI Halaman Registrasi.

Halaman registrasi digunakan untuk registrasi akun Terdapat terdapat 2 jenis *field* yaitu *user*name dan *password*. Selanjutnya terdapat tombol *submit* untuk *submitting* data.

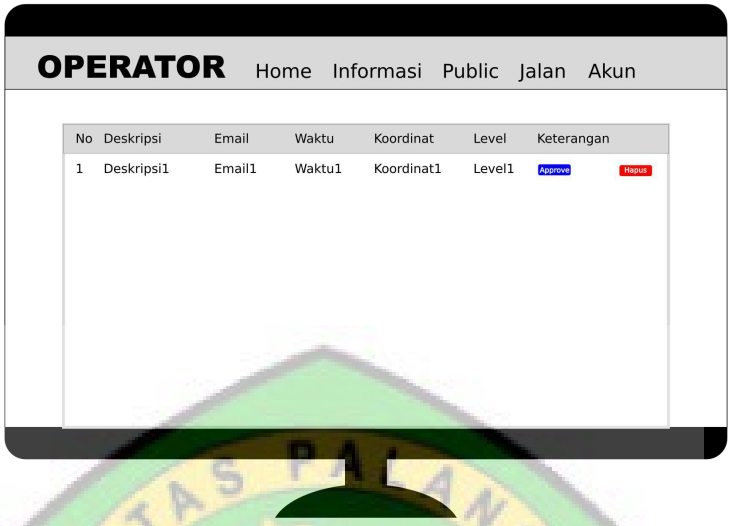


Gambar 3.24. Desain UI Halaman login.

Halaman login untuk autentikasi sebelum masuk ke dalam aplikasi sebagai *operator*. Sama halnya dengan halaman registrasi, terdapat terdapat 2 jenis *field* yaitu *user*name dan *password*. Selanjutnya terdapat tombol *submit* untuk *submitting* data.

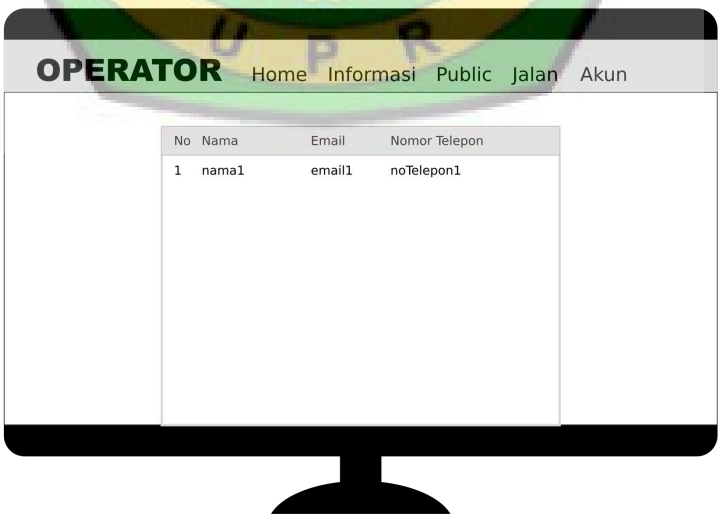
Gambar 3.25. Desain UI Halaman *Home*.

Halaman *home* merupakan *landing page* setelah *login*. Terdapat card jumlah informasi, 5 menu pada *menu bar*, yaitu *home*, informasi, *public*, jalan dan akun.



Gambar 3.26. Desain UI Halaman kelola informasi.

Halaman kelola informasi digunakan untuk mengelola informasi, yaitu berupa *approve* dan hapus informasi. Halaman ini berisi tabel dengan 8 kolom, yaitu nomor, deskripsi, *email*, waktu, koordinat, level, keterangan, serta kolom kosong untuk opsi hapus informasi. Jika terdapat informasi baru, maka pada kolom keterangan akan tersedia tombol *approve*.

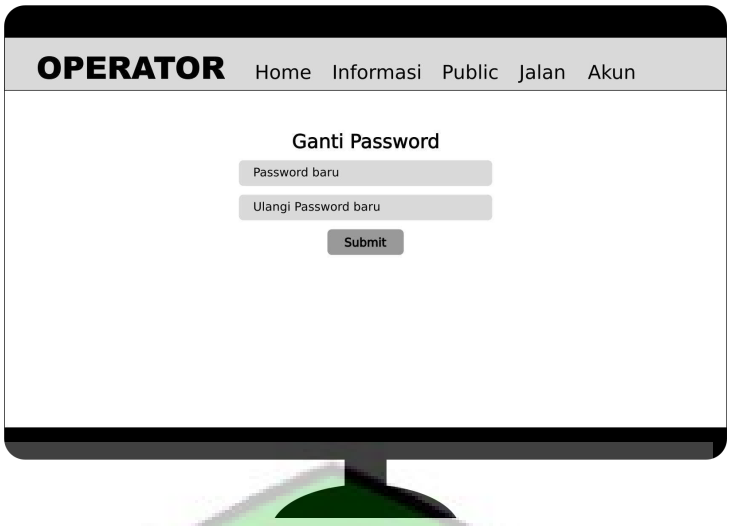


Gambar 3.27. Desain UI Halaman kelola *public*.

Kemudian pada halaman kelola *public* digunakan untuk mengelola data *public*, berupa hapus data *public*. Terdapat tabel dengan 4 kolom, yaitu nomor, nama, *email* dan nomor telepon.

Gambar 3.28. Desain UI Halaman kelola jalan.

Halaman kelola jalan digunakan untuk menambah koordinat atau titik persimpangan. Pada halaman ini berisi 2 bagian utama, yaitu bagian data *entry* dan bagian tabel. Pada bagian data *entry* terdapat 2 *field*, yaitu label dan koordinat, serta terdapat tombol *submit*. Bagian tabel berisi 3 kolom data, yaitu nomor, label dan koordinat.

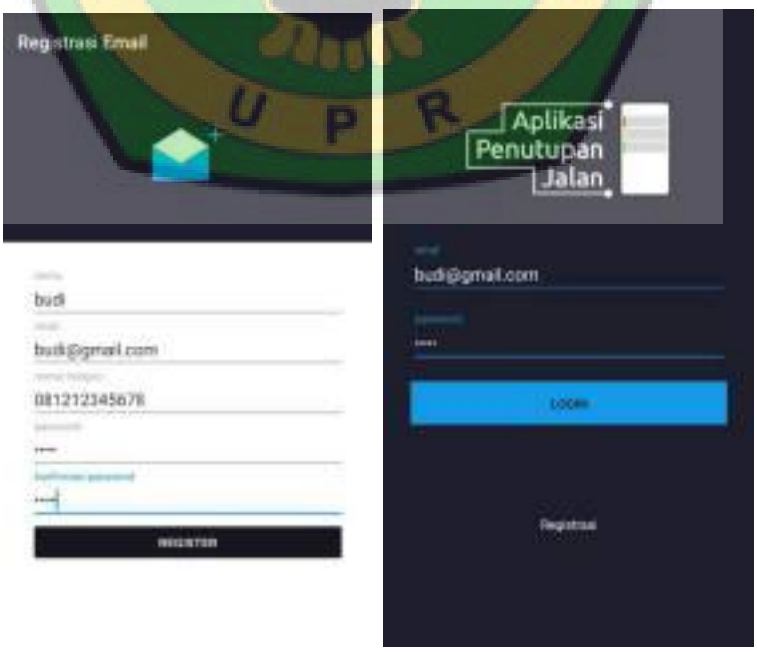


Gambar 3.29. Desain UI Halaman kelola akun.

Kemudian untuk halaman kelola akun digunakan utnuk mengganti *password*. Terdapat 2 *field*, yaitu *password* baru dan verifikasi *password* baru, serta tombol *submit*.

3.4.3. Implementasi dan Pengujian Iterasi Pertama

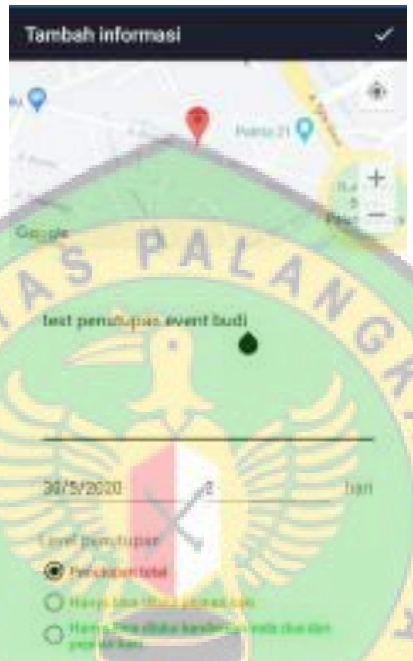
1) Registrasi dan *Login* pada aplikasi *mobile*



Gambar 3.30. Registrasi dan *login user public*.

user public melakukan registrasi untuk mendaftarkan *email* beserta data diri lainnya, kemudian melakukan *login* menggunakan *email* dan *password* yang telah didaftarkan sebelumnya.

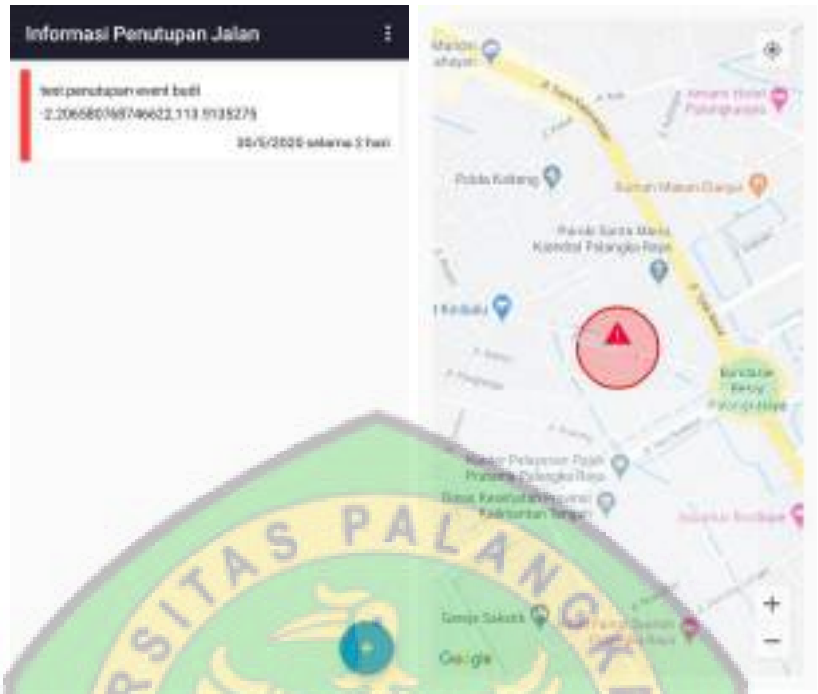
2) Membuat informasi penutupan jalan



Gambar 3.31. *user public* membuat informasi baru

user public malukukan penambahan informasi penutupan jalan, yaitu dengan memasukan deskripsi, tangal penutupan, lama penutupan dan level penutupan serta tidak lupa titik penutupan jalan pada *map View* yang tersedia.

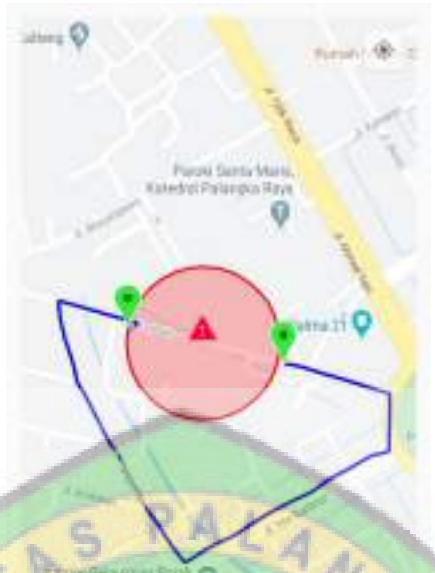
3) Melihat lokasi penutupan jalan pada *map*



Gambar 3.32. *View* titik penutupan jalan pada *map*.

User public menampilkan *map* dengan *view* yang langsung diarahkan pada titik penutupan jalan dengan cara menekan *card* informasi penutupan jalan yang ada pada daftar informasi penutupan jalan.

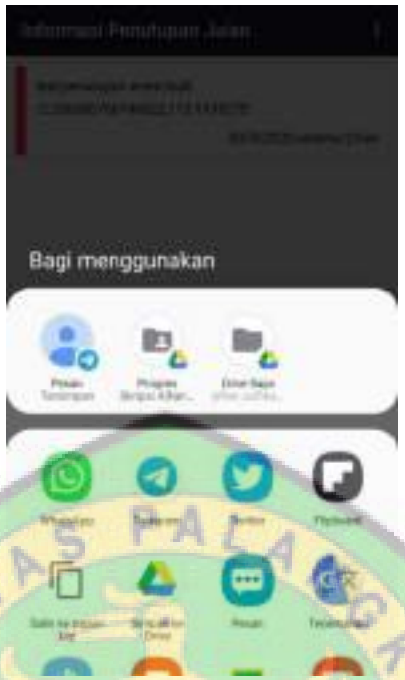
4) Melakukan *request* rute alternatif



Gambar 3.33. Rute hasil penerapan algoritma BFS dengan penambahan proses

User public melakukan *request* rute alternatif dengan memasukan titik asal dan tujuan. Titik asal dan titik tujuan ditentukan dengan menakan dan tahan (*long click*) pada titik atau daerah yang diinginkan.

5) Membagikan informasi via aplikasi eksternal

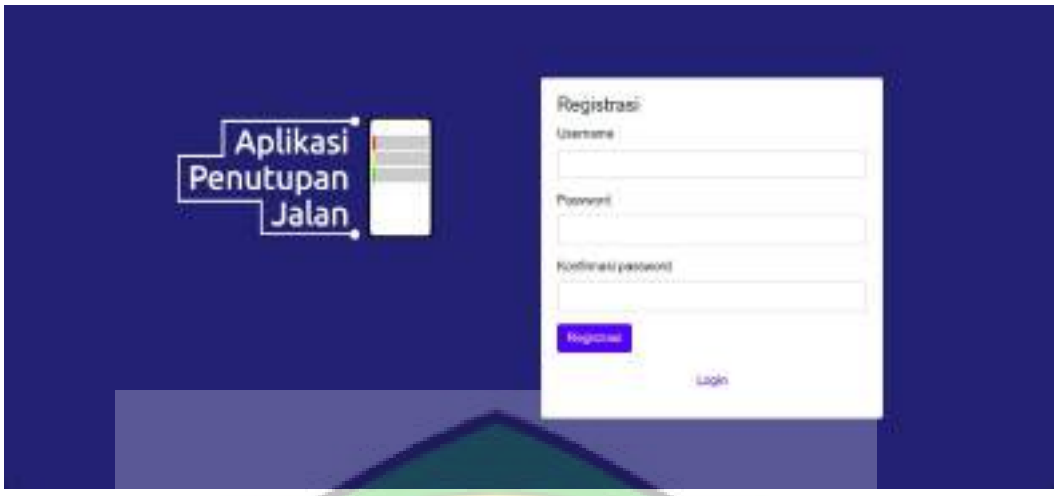


Gambar 3.34. Berbagi informasi via aplikasi eksternal.

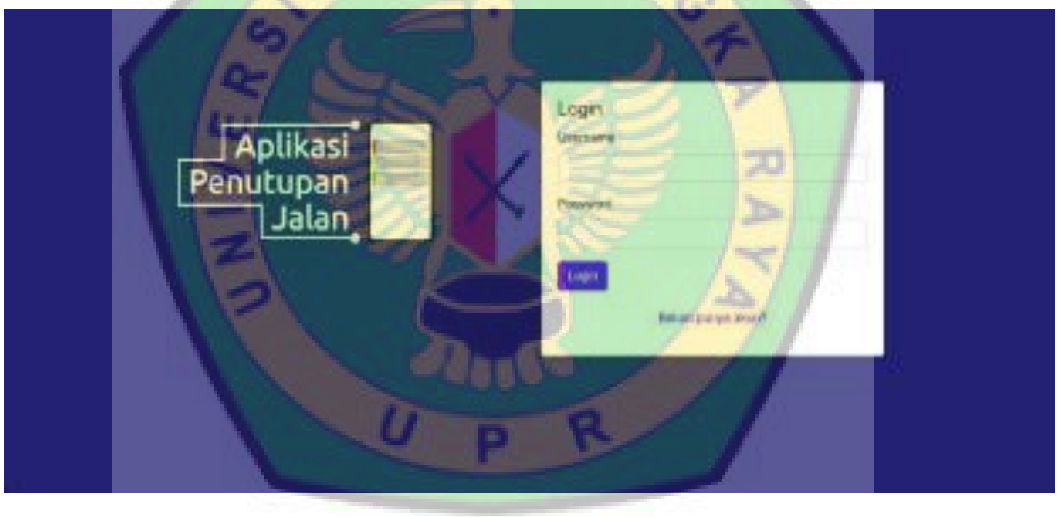
User public menekan dan tahan (*long click*) pada *card* informasi yang ingin dibagikan. Pada gambar tersebut terlihat aplikasi pada *slide* yang muncul. *user* dapat memilih aplikasi yang digunakan untuk berbagi informasi. Pada simulasi ini peneliti mencoba berbagi informasi via aplikasi *WhatsApp*. Maka hasilnya adalah sebagai berikut.

Gambar 3.35. Hasil *generate* informasi pada aplikasi *WhatsApp*.

6) Regitrasi dan *Login* pada aplikasi *website*

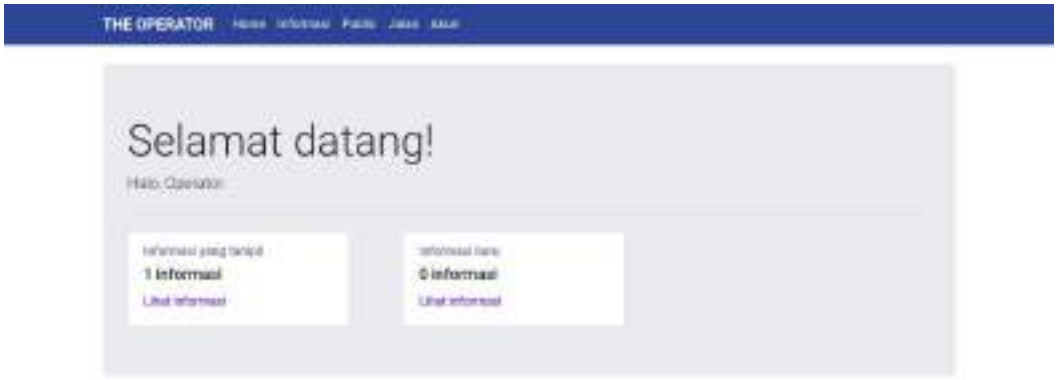


Gambar 3.36. Daftar informasi pada *website operator*.



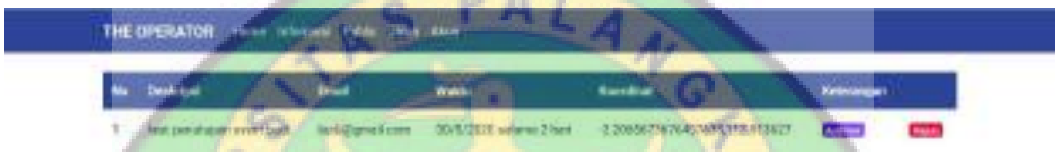
Gambar 3.37. Daftar informasi pada *website operator*.

Gambar 3.37 adalah halaman registrasi. *operator* melakukan registrasi untuk mendaftarkan *user*name dan *email*. Kemudian melakukan *login* untuk masuk kedalam *website* sebagai *operator*. Setelah berhasil, maka akan langsung berada pada halaman *home* berikut ini.



Gambar 3.38. Halaman *home*.

7) Mengelola informasi penutupan jalan



Gambar 3.39. Daftar informasi pada *website operator*.

Informasi yang baru dibuat oleh *user public* ditampilkan pada tabel kelola informasi *website operator*. Terdapat 2 pilihan bagi *operator*. Jika informasi ingin ditampilkan pada aplikasi *mobile* maka *operator* harus menekan tombol *approve*. Jika tidak, *operator* dapat menghapus informasi dengan menekan tombol hapus. Saat *operator* menekan tombol *approve*, maka data pada kolom keterangan berubah menjadi seperti pada gambar berikut ini.

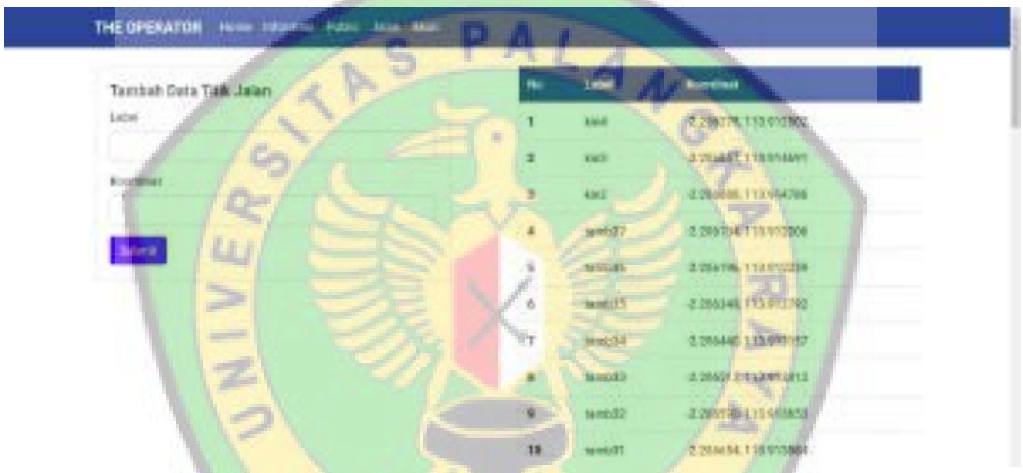
Gambar 3.40. Daftar informasi *website* setelah *approve* informasi.

8) Melihat data *user public*

Gambar 3.41. Melihat data *user public*.

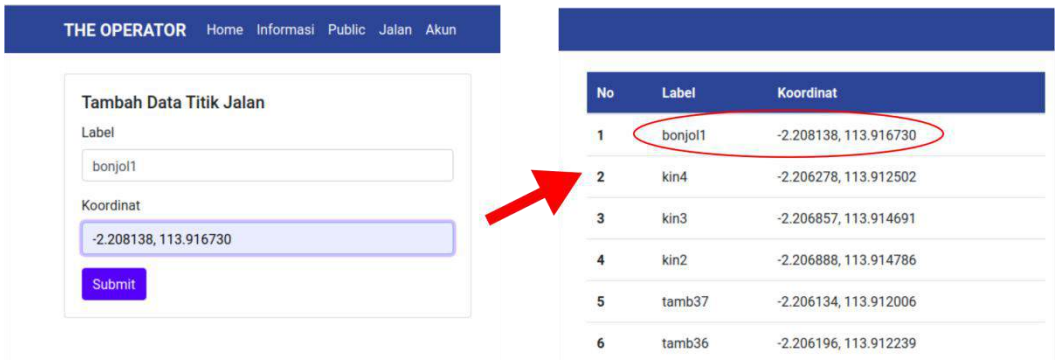
Pada halaman *public*, ditampilkan data *user public* yang terdaftar pada sistem. Data *user public* yang ditampilkan terdiri dari *email*, nama dan nomor telepon.

9) Menambah titik jalan



Gambar 3.42. Menambah titik jalan.

Pada halaman jalan, *operator* dapat menambah titik jalan. Tujuan dari fitur ini adalah untuk memperluas cakupan rute yang dapat di-*generate*. Berikut ini hasil simulasi dari penambahan titik jalan.



Gambar 3.43. Hasil penambahan titik jalan.

Untuk mengetahui apakah titik jalan yang baru *entry* sudah bisa diguanakan dalam pembuatan rute, dilakukan pencarian rute dengan asumsi bahwa titik asal dan tujuan berada disekitar titik jalan yang baru. Dan untuk memudahkan identifikasi titik jalan, peneliti menambahkan marker yang menandai titik jalan (hanya untuk simulasi).

Gambar 3.44. Simulasi pencarian rute melewati titik jalan yang baru.

Simulasi yang dilakukan pada gambar 3.44 adalah melakukan *request* rute dengan titik tujuan berada pada sekitar titik jalan baru (bonjol1). Terlihat rute yang dihasilkan berhasil melalui titik baru tersebut. Titik jalan baru ditunjukan pada panah merah. Titik jalan yang baru ditambahkan secara otomatis memiliki *Node* tetangga satu titik jalan yang paling dekat dengan titik jalan yang baru tersebut. Sistem melakukan pencarian terhadap satu titik jalan terdekat untuk dijadikan *Node* tetangga.

10) Mengganti *password* Akun *operator*

Gambar 3.45. Mengganti *password*.

Pada halaman akun, *operator* dapat mengganti *password* akun dengan memasukan *password* baru pada *field* yang tersedia pada *form*.

3.5. ITERASI KEDUA

Pada iterasi kedua, terdapat penambahan kebutuhan *user*. Kebutuhan *user* baru ditampilkan pada *user stories* berikut ini.

Tabel 3.8. *user Stories* iterasi kedua.

Judul	Deskripsi	Acceptance Criteria
Mendapatkan notifikasi peringatan penutupan jalan pada radius tertentu dari titik penutupan.	Sebagai *user public*, saya ingin mendapatkan peringatan penutupan jalan berupa notifikasi yang masuk ketika saya berada pada radius tertentu dari titik penutupan jalan.	Perangkat akan menerima notifikasi peringatan penutupan jalan saat berada pada radius 100 meter dari titik penutupan jalan.

Dari tabel tersebut, diketahui bahwa fitur yang akan ditambahkan adalah fitur peringatan informasi penutupan jalan berupa *push notification*. Fitur ini memanfaatkan sebuah *package* untuk melakukan *geofencing* atau pembatasan wilayah. Tahapan dari iterasi kedua ini adalah membuat *Planning* iterasi kedua, Implementasi program dan pengujian program.

3.5.1. Planning Iterasi Kedua

Berikut ini adalah *Planning* pengembangan sistem dari iterasi pengembangan sistem kedua.

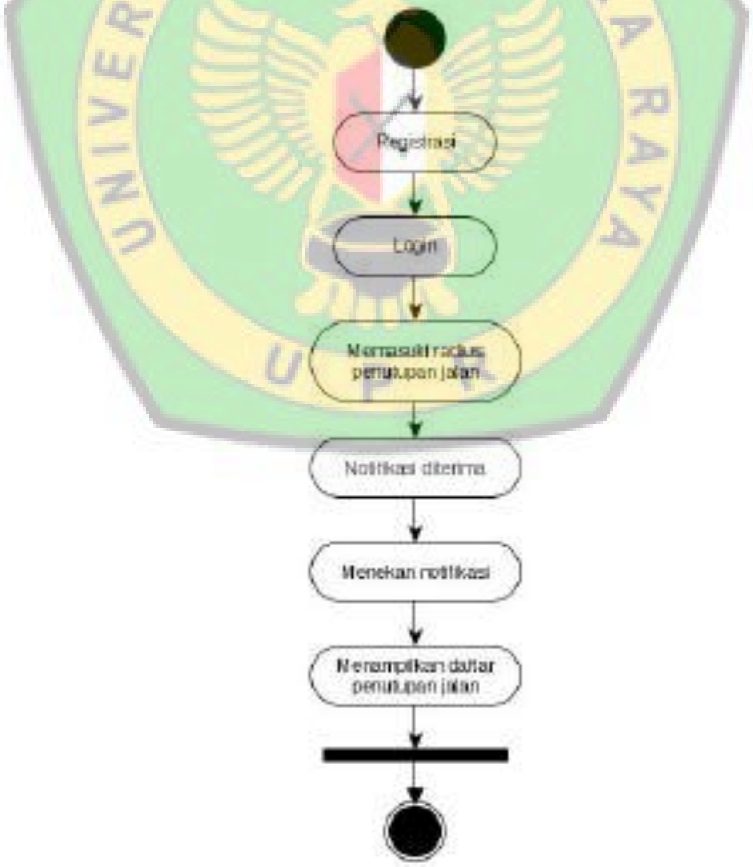
Tabel 3.9. *Planning* pengembangan sistem iterasi kedua.

Pengembangan Sistem	Acceptance Test
Mendapatkan notifikasi peringatan penutupan jalan pada radius tertentu dari titik penutupan.	*user public* mendapatkan peringatan penutupan jalan berupa notifikasi. Jika notifikasi ditekan akan mengarahkan *user* pada daftar penutupan jalan dalam keadaan aplikasi aktif atau berjalan di latar belakang.

3.5.2. Desain Sistem Iterasi Kedua

1) Activity diagram

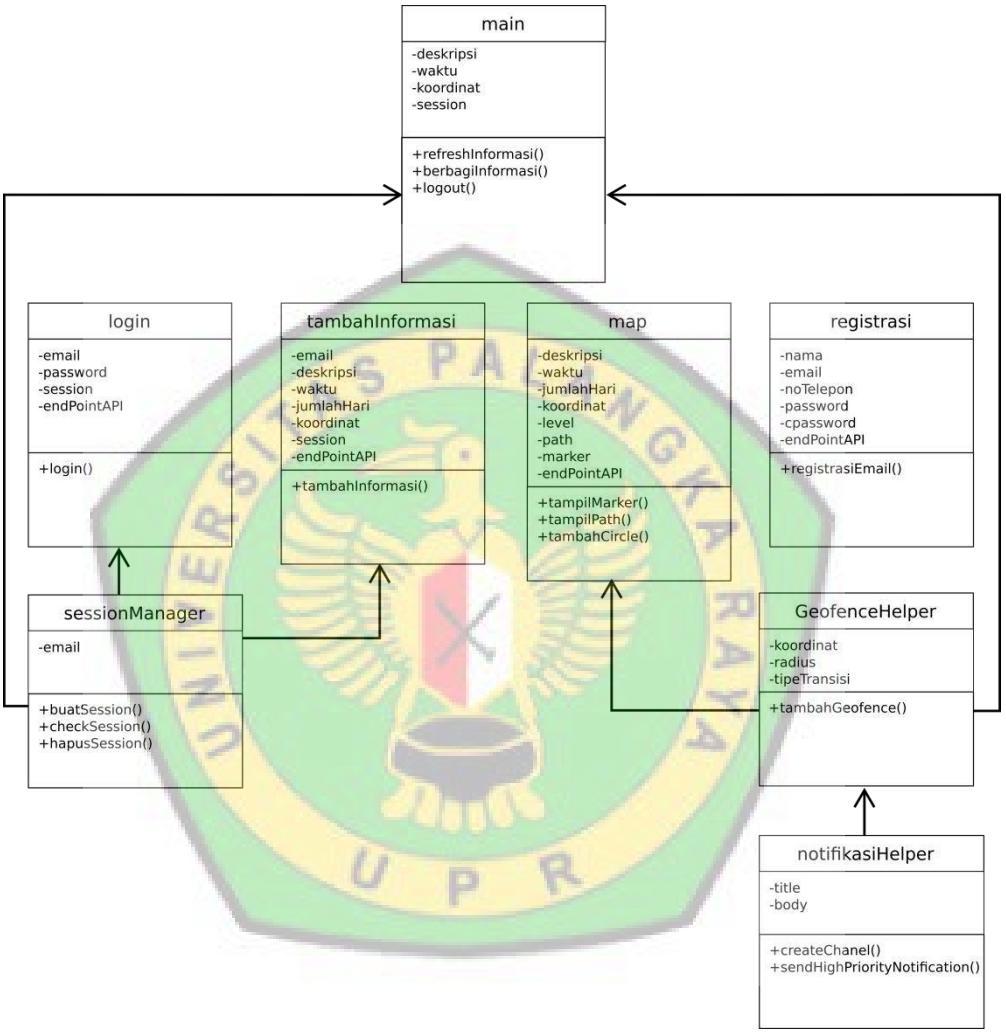
Berikut ini adalah activity diagram dari fitur yang akan ditambahkan bagi *user public* (aplikasi *mobile*), yaitu mendapatkan notifikasi peringatan penutupan jalan.



Gambar 3.46. Activity diagram *user public* pada iterasi kedua.

2) Class diagram

Dengan adanya penambahan fitur ini, maka *class diagram* pada aplikasi *mobile* juga berubah. Terdapat penambahan 2 kelas yang berhubungan dengan fungsi *geofence* dan *push notification*. Berikut ini *class diagram* dari aplikasi *mobile*.



Gambar 3.47. *Class diagram* aplikasi *mobile* pada iterasi kedua.

3.5.3. Implementasi dan Pengujian Iterasi Kedua

Terdapat 3 skenario pengujian, yaitu sebagai berikut.

1) Melintasi 1 titik penutupan jalan
2) Melintasi 2 titik penutupan jalan
3) *Force close* / menutup aplikasi (pengujian latar belakang)

Radius untuk geofence penutupan jalan diatur sejauh 100 meter. Berikut ini simulasi yang dilakukan.

1) Melintasi 1 titik penutupan jalan

Dibuat sebuah informasi penutupan jalan. Berikut ini informasi penutupan jalan yang dibuat.

Gambar 3.48. informasi penutupan jalan untuk pengujian geofence.

Selanjutnya peneliti melintasi wilayah tersebut. Pada saat mendekati wilayah penutupan jalan, maka muncul notifikasi seperti berikut ini.

Gambar 3.49. Notifikasi peringatan penutupan jalan.

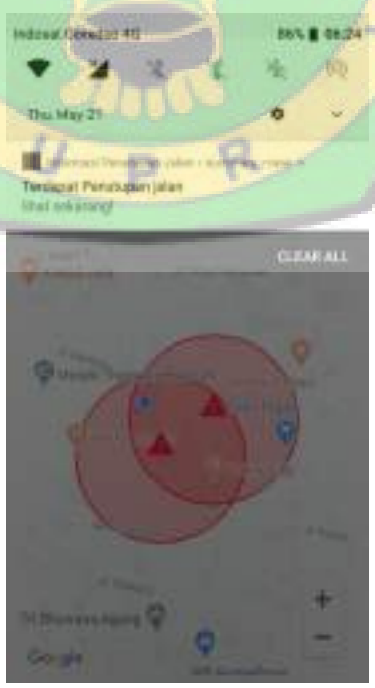
Didapatkan notifikasi peringatan penutupan jalan yang ada di sekitar lokasi perangkat. Ini menunjukan bahwa *geofence* sudah bekerja.

2) Melintasi 2 titik penutupan jalan

Selanjutnya peneliti membuat informasi penutupan jalan kedua yang berdekatan dengan informasi penutupan jalan yang pertama.

Gambar 3.50. 2 informasi penutupan jalan.

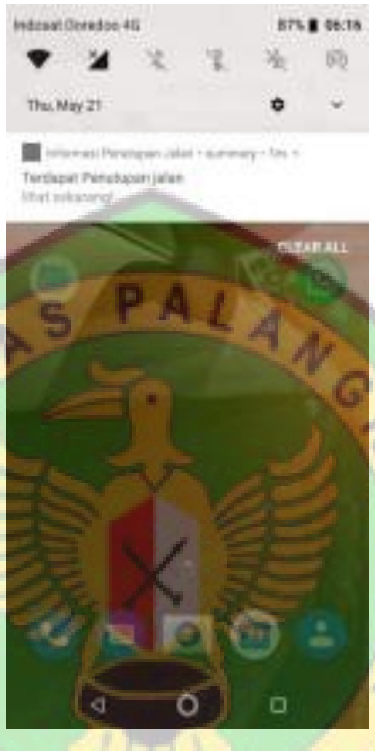
Peneliti mendekati titik penutupan jalan kedua. Hasilnya adalah muncul notifikasi peringatan penutupan jalan lainnya.



Gambar 3.51. Notifikasi peringatan penutupan jalan.

3) *Force close* / menutup aplikasi (pengujian latar belakang)

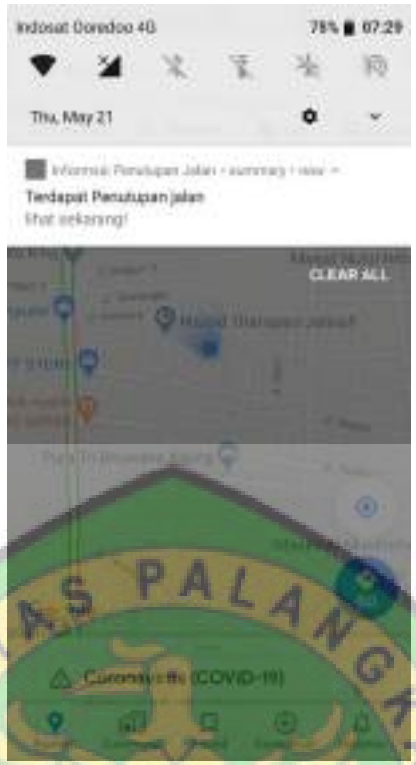
Aplikasi di-*force close* atau ditutup. Kemudian peneliti mendekati titik penutupan jalan. Setelah beberapa menit *roaming* disekitar wilayah titik penutupan jalan, maka muncul notifikasi peringatan penutupan jalan seperti berikut ini.



Gambar 3.52. Notifikasi peringatan penutupan jalan saat aplikasi ditutup.

Peneliti mendapati bahwa notifikasi didapatkan dalam waktu yang lebih lama jika dibandingkan pada saat aplikasi aktif.

Kemudian peneliti memanfaatkan aplikasi eksternal yang melakukan *request* lokasi untuk menguji notifikasi. Maka digunakanlah G*oogle maps*. Pada saat peneliti melintasi titik penutupan jalan dan *Google maps* dibuka, maka seketika notifikasi muncul seperti pada gambar berikut ini.



Gambar 3.53. Notifikasi peringatan penutupan jalan saat pengujian dengan membuka aplikasi *Google maps*.

Saat membuka aplikasi *Google maps*, perangkat langsung menerima notifikasi peringatan penutupan jalan yang disebabkan adanya *request current location* pada perangkat saat menggunakan aplikasi *Google maps*.

BAB IV
HASIL DAN PEMBAHASAN

4.1. SMALL RELEASE

Setelah dilakukan 2 kali iterasi pengembangan, didapatlah sistem yang sudah memenuhi semua kebutuhan *user*. Berikut ini fungsionalitas sistem yang telah selesai dibuat.

4.1.1. Fungsionalitas Apilikasi *mobile*

Tabel 4.1. Fungsionalitas Aplikasi *mobile*.

Fungsi	Keterangan
Melakukan registrasi dan login pada aplikasi *mobile*.	*User* melakukan registrasi untuk mendaftarkan email serta data diri lain seperti nama dan nomor telepon. Kemudian email yang telah didaftarkan dapat digunakan untuk login pada aplikasi.
Membuat informasi penutupan jalan	Informasi dibuat oleh *user* Public, kemudian Operator dapat melakukan *approve* atau hapus informasi yang baru dibuat oleh *user* Public. Data yang di-*entry* anatara lain deskripsi informasi, tanggal penutupan, lama penutupan, level penutupan dan titik pentupan jalan.
Melihat lokasi penutupan pada *Map*.	Untuk melihat titik penutupan pada *Map*, *user Public* menekan *card* informasi yang ingin dilihat, maka akan muncul tampilan map dengan *view* yang langsung diarahkan pada titik penutupan tersebut.

Tabel 4.1. (lanjutan).

Fungsi	Keterangan
Melakukan *request* rute alternatif.	Untuk mendapatkan rute, *user Public* harus memasukan titik asal dan tujuan, dengan menekan dan tahan (*long click*) pada *Map*. Setelah memasukan titik kedua (tujuan) maka rute otomatis ditampilkan. Untuk mendapat rute dengan asal dan tujuan yang lain, *user Public* dapat langsung melakukan hal yang sama. Otomatis rute akan ter-*reset*.
Membagikan informasi via aplikasi eksternal.	*User public* dapat berbagi informasi penutupan jalan tertentu menggunakan beberapa aplikasi *messenger* yang terinstal di perangkat *user*. Caranya dengan menekan dan tahan (*long click*) pada *card* informasi yang ingin dibagikan. Kemudian akan muncul *slide* yang menyediakan pilihan aplikasi yang ingin digunakan. Setelah *user Public* memilih aplikasi yang akan digunakan, informasi otomatis ter-*generate* menjadi teks yang terdiri dari deskripsi penutupan, waktu penutupan, level penutupan dan link *Google Maps*.

Tabel 4.1. (lanjutan).

Fungsi	Keterangan
Mendapatkan notifikasi peringatan penutupan jalan pada radius tertentu dari titik penutupan.	Jika pada suatu wilayah terdapat penutupan jalan dan *user* berada pada radius kurang dari 100 meter dari titik pentupan jalan, maka akan muncul notifikasi berisi peringatan penutupan jalan. *Catatan : akurasi penerimaan notifikasi akan lebih baik jika terdapat aplikasi yang melakukan *request* lokasi seperti aplikasi ini (apk penutupan jalan), *Google Maps*, dan sejenisnya. Jika tidak, maka notifikasi akan diterima cukup lama.

4.1.2. Fungsionalitas Aplikasi Website

Tabel 4.2. Fungsionalitas Aplikasi Website.

Fungsi	Keterangan
Melakukan registrasi dan *login* pada aplikasi website.	*Operator* melakukan registrasi untuk mendapatkan *user*name dan *Password*. Kemudian *user*name dan *password* tersebut dapat digunakan untuk login pada aplikasi.

Tabel 4.2. (lanjutan).

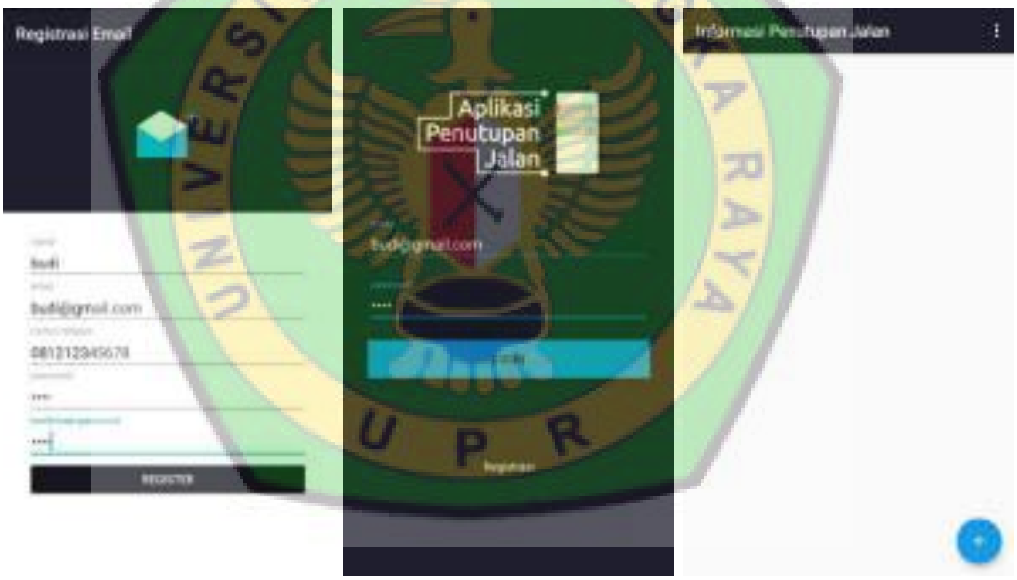
Fungsi	Keterangan
Mengelola informasi penutupan jalan.	Informasi yang dibuat oleh *user Public* dikelola oleh Operator pada halaman informasi. Pada tabel tersebut disajikan data informasi yang ditampilkan pada aplikasi *mobile* ataupun informasi yang baru masuk. Data informasi yang ditampilkan antara lain deskripsi informasi, email *user Public* yang membuat informasi, koordinat lokasi penutupan, level penutupan dan keterangan penutupan. *Operator* dapat *approve* atau menghapus informasi.
Melihat data *user Public*.	Untuk melihat data *user Public*, *Operator* menuju halaman *public*. Terdapat tabel yang menyajikan data *user Public*.
Menambah titik jalan.	*Operator* dapat menambahkan titik jalan dengan masuk pada aplikasi website halaman kelola jalan, kemudian memasukan label persimpangan serta koordinatnya. Koordinat persimpangan bisa didapatkan dari perangkat GPS ataupun menggunakan *Google Maps*. Setelah operator melakukan submit, aplikasi secara otomatis menambahkan titik jalan baru.
Mengganti *password* Akun *operator*.	*Operator* dapat mengganti *password* akun dengan menuju halaman akun. Tersedia *form* untuk mengganti *password* dengan.

4.2. PEMBAHASAN

Berikut ini adalah pembahasan dari pengembangan sistem yang telah dilakukan. Terdapat 3 hal yang akan dibahas, yaitu mekanisme pembuatan informasi penutupan jalan, penerapan *Location Based Servce* dalam pembuatan informasi penutupan jalan dan peringatan penutupan jalan, serta penerapan Graf dalam pembuatan rute.

4.2.1. Mekanisme Pembuatan Informasi Penutupan Jalan

Mengacu pada mekanisme pembuatan informasi pembuatan jalan seperti yang disajikan pada gambar 3.12 (Bab 3), untuk membuat informasi penutupan jalan *user* Public harus melakukan registrasi agar bisa login ke dalam aplikasi *mobile*.



Gambar 4.1. *user Public* melakukan registrasi dan *login*.

Kemudian *user public* menuju halaman tambah informasi. Pada *form* yang tersedia, *user public* melakukan *entry* data informasi penutupan jalan.

Gambar 4.2. *user* mengisi data informasi penutupan jalan.

Data tersebut diantaranya deskripsi penutupan jalan, tanggal penutupan, lama penutupan, level penutupan dan titik penutupan jalan. Kemudian *user public* melakukan submit data dengan menekan tombol centang, sehingga informasi yang baru dibuat tadi masuk ke dalam *database*.

Informasi yang ditampilkan pada daftar informasi aplikasi *mobile* hanya informasi yang memiliki keterangan *approved*. Berikut ini tampilan dashboard *database* manager saat informasi baru saja ditampilan. Peneliti menggunakan *phpmyadmin* dari *xampp*.

Gambar 4.3. Informasi baru pada *database manager*.

Terlihat pada kolom keterangan, nilainya adalah "*waiting*". Nilai ini hasil dari *generate* otomotis setelah *user public* melakukan *submit* informasi baru.

Gambar 4.4. Informasi yang baru dimasukan.

Kemudian pada halaman kelola informasi *website operator* terdapat tombol biru dengan tulisan *approve*. Tombol ini aktif dengan sendirinya pada setiap informasi yang baru saja masuk. Jika *operator* menekan tombol tersebut, keterangan akan berubah otomatis menjadi "*approved*" seperti pada gambar berikut ini.

Gambar 4.5. Informasi baru setelah di-*approve*.

Kemudian *user public* dapat me-*refresh* daftar informasi. Sehingga informasi baru akan ditampilkan.

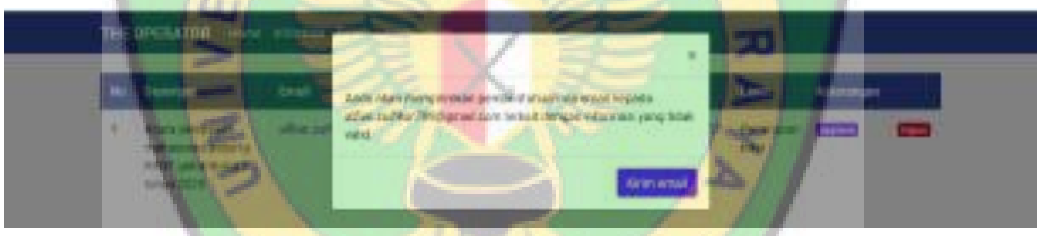
Gambar 4.6. Informasi baru ditampilkan pada daftar informasi.

Informasi yang tampil pada daftar informasi pada aplikasi *mobile* akan terhapus secara otomatis jika sudah melewati waktu yang tertera. Seperti pada gambar 4.6, terlihat informasi penutupan jalan terjadi pada tanggal 30 Mei 2020 selama 2 hari, artinya penutupan jalan akan terjadi sampai tanggal 31 Mei 2020. Setelah itu informasi akan hilang dengan sendirinya. Jika kembali pada halaman kelola

informasi *website* (Gambar 4.7), keterangan berubah menjadi *expired*, menandakan bahwa informasi sudah tidak ditampilkan.

Gambar 4.7. Informasi telah *expired*.

Deskripsi informasi penutupan jalan harus disertai dengan keterangan jalan atau lokasi dari penutupan jalan. Berkaitan dengan validasinya, *operator* dimungkinkan untuk mengirimkan *email* kepada *user public*, dimana isi *email* tersebut adalah pemberitahuan bahwa informasi yang baru dimasukkan tidak valid dan harus menyertakan nama jalan dan keterangan lainnya. *Operator* hanya tinggal menekan *email* dari *user* tersebut, kemudian muncul jendela *pop up* untuk men-*generate* dan mengirim *email*.



Gambar 4.8. Jendela *pop up* untuk mengirim *email* kepada *user public*.

Selanjutnya *operator* menekan tombol "kirim email" Untuk mengirimkan *email* menuju *user public* sesuai dengan *email* yang tertera. Berikut ini email yang berhasil dikirimkan kepada *user public*.

Gambar 4.9. *Email* yang berhasil dikirim.

4.2.2. Penerapan Location Based Service dalam Pembuatan informasi dan Peringatan Penutupan Jalan

Teknologi LBS adalah teknologi yang memanfaatkan data lokasi perangkat untuk diolah menjadi informasi. Pelacakan lokasi perangkat bisa didapatkan dari pelacakan GPS atau jaringan yang disediakan *provider*. Pada penelitian ini, data lokasi perangkat didapatkan dari pelacakan GPS. Untuk memperoleh data lokasi perangkat (perangkat *mobile*), peneliti menambahkan *permission* untuk bisa menggunakan atau mengakses layanan *current location* dari perangkat yang digunakan oleh *user*. Berikut ini deklarasi *permission* pada file manifest.

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
```

Gambar 4.10. Pendeklarasian *permission* untuk mengakses data lokasi perangkat.

Selain pelacakan lokasi perangkat, sistem yang dibuat juga mampu mengambil koordinat sebuah titik di wilayah penelitian. Peneliti menggunakan layanan dari *Google Maps*, yaitu *Google Maps* API untuk mendapatkan akses *view map* dari *Google*. Ini digunakan untuk menentukan lokasi penutupan jalan yang nantinya disertakan dalam informasi yang dibuat seperti gambar berikut ini.

Gambar 4.11. Pemanfaat data lokasi untuk informasi pentupan jalan.

Sistem yang dibuat dilengkapi dengan fitur peringatan penutupan jalan berupa noifikasi yang dapat diterima perangkat *mobile*. Untuk membuat fitur peringatan penutupan jalan, peneliti menerapkan *class-class* yang berhubungan dengan geofence pada package "com.google.android.gms.location". *Geofence* atau pembatasan wilayah bekerja dengan menandai suatu koordinat lokasi dengan radius tertentu dan memberikan *trigger* kepada perangkat ketika terjadi transisi.

Terdapat 3 transisi pada *geofence*, yaitu *enter*, *dwell* dan *exit*. Namun transisi yang diterapkan pada program adalah hanya transisi *enter*, yaitu kondisi geofence mendapatkan *trigger* ketika perangkat memasuki daerah radius lokasi. Sementara untuk transisi *dwell* dan *exit* tidak digunakan. Trasisi *dwell* adalah kondisi dimana perangkat berjelajah di dalam daerah radius lokasi beberapa saat, sementara transisi *exit* adalah kondisi dimana perangkat keluar dari daerah radius lokasi.

Dalam hal ini, peneliti menyematkan radius 100 meter pada setiap titik penutupan jalan. Sehingga ketika perangkat berada pada radius kurang dari 100 meter dari titik penutupan jalan, akan dibuat sebuah notifikasi yang memberi tahu jika terdapat penutupan jalan. Percobaan telah dilakukan pada saat pengembangan sistem (dapat dilihat pada Bab 3 bagian iterasi kedua). Percobaan dilakukan dengan 3 skenario, yaitu melewati 1 penutupan jalan, melewati 2 penutupan jalan yang berdekatan dan *force close* atau menutup aplikasi.

Dari skenario ketiga, diketahui bahwa aplikasi beberapa kali berhasil memberikan notifikasi peringatan penutupan jalan. Namun aplikasi memberikan notifikasi terlalu lama dan bahkan harus mengaktifkan aplikasi yang melakukan *request current location* sepert *google maps* agar aplikasi memberikan notifikasi. Ini disebabkan karena fungsi *geofence* akan bekerja lebih baik jika perangkat yang digunakan melakukan request lokasi. Sehingga pada saat aplikasi ditutup dan tidak ada aplikasi lain yang melakukan request lokasi, notifikasi sulit didapatkan. Peneliti menyimpulkan notifikasi yang diberikan oleh aplikasi penutupan jalan ini saat berjalan di latar belakang (*background service*) tidak seakurat dan seaktual pada saat kondisi aplikasi aktif. Hal ini diperkuat dengan percobaan dengan menggunakan aplikasi *Google Maps* yang mana aplikasi *Google Maps* melakukan request lokasi, sesaat setelahnya notifikasi langsung muncul.

4.2.3. Penerapan Graf dalam Pembuatan Rute

Data yang diolah dalam pengimplementasian graf ini adalah data koordinat titik persimpangan yang disimpan pada *database* tabel persimpangan. Berikut ini tampilan tabel pada *phpmyadmin*.

	idPersimpangan	label	lating	nodeTetangga
Edit Copy Delete	1	yos1	-2.207654, 113.915717	tamb1,kin1
Edit Copy Delete	2	yos2	-2.209156, 113.913311	tamb11,tamb12
Edit Copy Delete	3	yos3	-2.211053, 113.910068	yos4,hend9
Edit Copy Delete	4	yos4	-2.211471, 113.909309	yos3
Edit Copy Delete	5	kra3	-2.206505, 113.912852	tamb13,tamb14
Edit Copy Delete	6	hend9	-2.209778, 113.909444	yos3
Edit Copy Delete	7	kin1	-2.207175, 113.915728	yos1,tamb24
Edit Copy Delete	8	tamb1	-2.207900, 113.915505	yos1,tamb2
Edit Copy Delete	9	tamb2	-2.208061, 113.915231	tamb3,tamb1

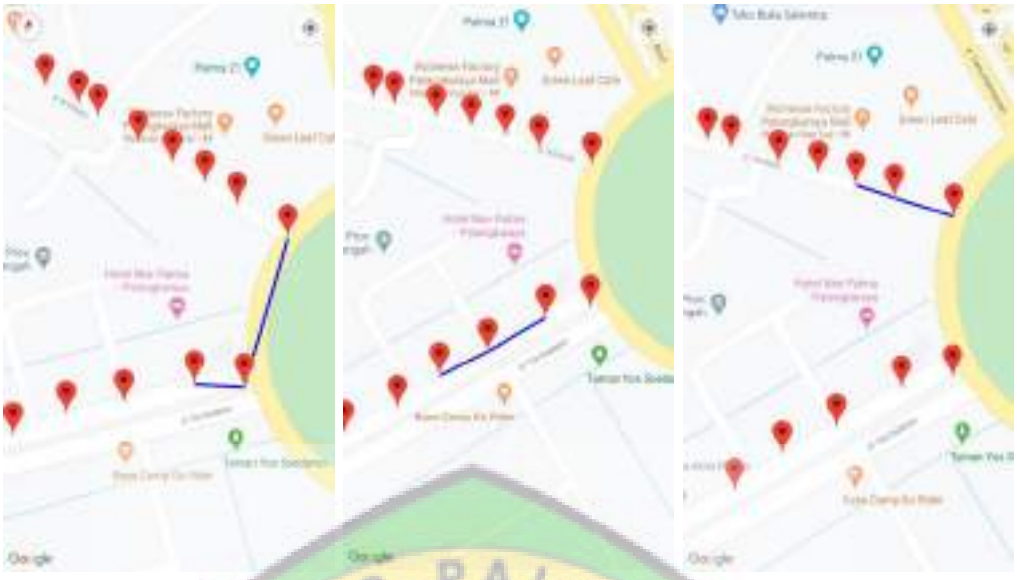
Gambar 4.12. Tabel persimpangan.

Tabel persimpangan memiliki 4 kolom, yaitu idPersimpangan, label, koordinat dan nodeTetangga. Setiap titik persimpangan memiliki *node* tetangga, dimana *node* tetangga adalah titik persimpangan yang berhubungan langsung dengan suatu *node* yang bertetangga padanya. Inilah yang coba dimanfaatkan untuk kemudian data pada tabel tersebut diimplementasikan sebagai graf untuk membuat rute.

Untuk menjelaskan bagaimana setiap data yang ada pada tabel persimpangan (gambar 4.12) saling berhubungan sesuai dengan *node* tetangganya, peneliti membuat program sederhana, dimana semua data pada tabel persimpangan akan di-*load* dan ditampilkan pada *Map*. Berikut ini tampilan program yang dibuat.

Gambar 4.13. Tampilan program untuk menampilkan data titik jalan.

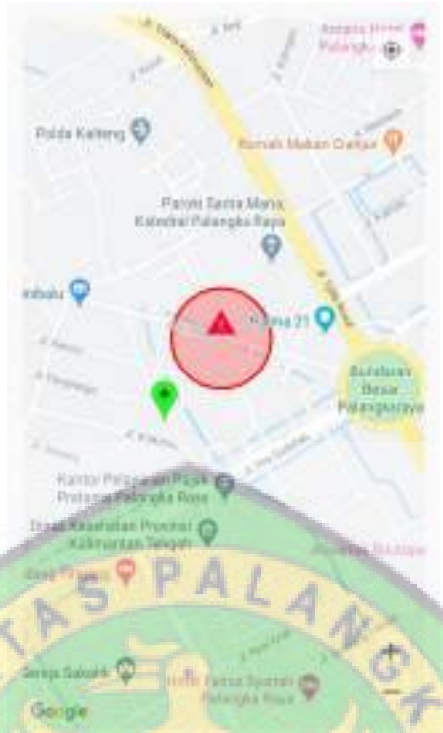
Kemudian ditambahkan fungsi untuk membuat *polyline* yang menghubungkan *node* dengan *node* tetangganya. *Polyline* akan terbentuk saat *marker* ditekan (*click*). Ketika marker ditekan, akan terbentuk *polyline* yang menghubungkan *marker* (*node*) dengan *marker* tetangganya (*node* tetangga). berikut ini simulasinya.



Gambar 4.14. Simulasi menampilkan *polyline* yang menghubungkan *node* dengan *node* tetangganya.

Dari simulasi tersebut, terlihat bahwa titik jalan yang tersimpan di tabel persimpangan dapat terhubung dengan titik jalan yang bertetangga dengannya. Sehingga terbentuklah sebuah struktur Graf tidak berarah yang tersusun dari data titik jalan yang saling terhubung satu sama lain sesuai dengan *node* tetangganya. Struktur Graf tidak berarah ini diasumsikan sebagai struktur jalan.

Cara ini juga yang dilakukan untuk memvisualkan rute pada aplikasi, yaitu membentuk *polyline* yang menghubungkan satu titik jalan dengan titik jalan yang lain. Titik jalan penyusun rute didapatkan dari kalkukasi oleh sistem dengan menerapkan sebuah *rules*. Untuk mendapatkan rute, *user* public harus menentukan titik asal dan tujuan. *user* public membuka tampilan map. Kemudian melakukan klik dan tahan (*long click*) untuk menandai titik asal dan tujuan.



Gambar 4.15. Memasukan titik asal dan tujuan.

Pada saat melakukan *long click* pertama kali, *marker* hijau yang muncul merupakan titik asal. Kemudian melakukan *long click* yang kedua untuk memunculkan *marker* sebagai titik tujuan. Rute akan ter-*generate* secara otomatis setelah melakukan *long click* yang kedua atau saat menentukan titik tujuan. Berikut ini tampilan rute pada *map*.

Gambar 4.16. Hasil *request* rute.

Jika memperhatikan gambar 4.16, terlihat garis biru yang menghubungkan marker asal dan tujuan (marker berwarna hjau). Pada dasarnya, rute terbentuk dari *polyline* yang disusun satu persatu antara satu titik jalan dengan titik jalan lain, dimana titik jalan penyusun rute didapatkan dari hasil kalkulasi pencarian rute oleh sistem. Berikut ini adalah tampilan rute jika titik penyusun rute ditampilkan sebagai *marker*.

Gambar 4.17. *Marker* titik jalan penyusun rute.

Terdapat 2 proses besar yang terjadi sampai rute ditampilkan, yaitu sebagai berikut.

a. Penambahan titik asal, titik tujan dan titik penutupan jalan kedalam Graf.

Proses penambahan ini bertujuan untuk menghubungkan titik asal, titi tujuan dan titik penutupan jalan dengan struktur Graf yang terdiri dari titik jalan yang ada pada *database*. Penelusuran rute baru dapat dilakukan jika ketiga jenis titik tadi telah terhubung dengan titik jalan yang ada pada *database*. Cara yang dilakukan peneliti adalah dengan mencari tahu titik jalan terdekat dengan titik asal, titik tujuan dan titik penutupan jalan. Pada penelitian ini digunakan rumus *haversine* untuk menghitung jarak antara dua titik pada permukaan bumi dengan nilai koordinat (garis lintang garis bujur / *latitude longitude*) sebagai variabelnya. Rumus dari perhitungan tersebut adalah sebagai berikut.

$$d = 2r \arcsin\left(\sqrt{\operatorname{hav}(\varphi_2 - \varphi_1) + \cos(\varphi_1)\cos(\varphi_2)\operatorname{hav}(\lambda_2 - \lambda_1)}\right)$$
$$= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\varphi_2 - \varphi_1}{2}\right) + \cos(\varphi_1)\cos(\varphi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Gambar 4.18. Rumus *Haversine*. (Wikipedia)

Rumus ini menghasilkan keluaran berupa nilai jarak antara dua titik koordinat. Ini lah yang coba dimanfaatkan peniliti untuk mencari titik jalan terdekat dari titik asal, titik tujuan dan titik penutupan jalan. Perhitungan ini dilakukan terhadap semua titik jalan yang ada pada *database*. Setelah mengetahui titik jalan mana yang terdekat, barulah titik jalan terdekat ini dijadikan sebagai *node* tetangga dari masing-masing titik asal, titik tujuan dan titik penutupan jalan. Perhitungan ini juga digunakan dalam fitur penambahan titik jalan (Bab 3) untuk menentukan *node* tetangga dari titik jalan yang baru ditambahkan.

Berikut ini keseluruhan proses penambahan titik asal, titik tujuan dan titik penutupan jalan ke dalam Graf yang terjadi pada sistem.

1) Sistem mengambil titik koordinat asal, tujuan, dan semua penutupan jalan.
2) Sistem me-*load* data titik jalan dari *database* (tabel persimpangan).
3) Sistem melakukan kalkulasi jarak antara titik koordinat asal, tujuan dan semua penutupan jalan dengan koordinat titik jalan yang tersimpan di *database*.
4) Sistem mendapatkan titik jalan yang memiliki jarak paling dekat dengan masing-masing titik asal, tujuan dan semua penutupan jalan.
5) Titik asal, tujuan dan semua penutupan jalan dihubungkan dengan masing-masing titik jalan yang memiliki jarak terdekat, sehingga tersambung dengan struktur Graf yang sudah ada.

b. Pencarian rute.

Setelah melakukan proses penambahan titik asal, titik tujuan dan titik penutupan jalan pada Graf, barulah dilakukan kalkulasi pencarian rute. Diasumsikan bahwa sudah tersedia Graf yang akan digunakan untuk penelusuran rute. Algoritmanya adalah sebagai berikut.

1) Memasukan *node* ujung (titik asal) ke dalam sebuah antrian.
2) Dilakukan pemeriksaan semua titik yang ada pada antrian.
3) Jika *node* yang diperiksa merupakan *node* hambatan (titik penutupan), maka *node* tersebut dan *node* tetangganya diabaikan dan kembali pada poin (2). Jika *node* yang diperiksa bukan *node* hambatan, maka lanjut ke poin (4).
4) Jika saat dilakukan pemeriksaan ditemukan solusi (titik tujuan), hasil dikembalikan dan selesai. Jika bukan solusi, lanjut ke poin (5).
5) Dilakukan pemeriksaan apakah *node* memiliki tetangga. Jika *node* memiliki tetangga, semua *node* tetangga dimasukan kedalam antrian dan kembali ke poin (2). Jika tidak, hasil tidak ditemukan dan selesai.

Dari akgoritma tersebut, sistem mendapatkan titik-titik jalan yang menyusun rute. Selanjutnya sistem membuat *polyline* (garis lurus) yang menghubungkan koordinat titik jalan yang menyusun rute dari titik asal hingga titik tujuan, sehingga terbentuklah rute

Algoritma atau *rules* yang digunakan untuk menentukan rute alternatif dibuat dengan memodifikasi agoritma *Breadth First Search* (BFS). Rules dapat dilihat pada gambar 3.7 (Bab 3 bagian metode pengolahan data). Untuk mengetahui perbedaan penggunaan algoritma BFS reguler dengan *rules* yang digunakan dalam sistem ini, maka dilakukan pengujian pencarian rute dengan dua kondisi. Kondisi pertama adalah dengan menerapkan algoritma BFS, dan kondisi kedua adalah dengan menambahkan proses pengecekan titik hambatan. Berikut ini rute yang dihasilkan pada pengujian pencarian rute terhadap kedua kondisi tersebut.

Gambar 4.19. Rute hasil penerapan algoritma BFS reguler/biasa.

Gambar 4.20. Rute hasil penerapan algoritma BFS dengan penambahan proses.

Dari kedua gambar diatas, terlihat bahwa algoritma BFS reguler tidak membaca titik penutupan jalan sebagai hambatan yang harus diabaikan dalam pencarian jalur. Hal ini menyebabkan rute yang dihasilkan tetap melibatkan titik penutupan jalan sebagai bagian rute yang dipilih. Sedangakan pada pengujian pada algoritma BFS dengan penambahan proses pengecekan *node* hambatan, terlihat bahwa rute yang didapatkan tidak melalui titik penutupan jalan. Perbedaan algoritma BFS reguler dan *rules* pencarian rute yang digunakan pada sistem ini dapat dilihat pada *flowchart* gambar 3.7 dan 3.8 (Bab 3 bagian metode pengolahan data).

BAB V
KESIMPULAN DAN SARAN

5.1. KESIMPULAN

Dalam penelitian skripsi ini, dilakukan pembuatan Aplikasi Penutupan Jalan dengan menerapkan teknologi *Location Based Service* (LBS) dan struktur data Graf. Berkaitan dengan LBS, dilakukan pemanfaatan data lokasi perangkat (*mobile*) untuk membuat informasi penutupan jalan. Sedangkan untuk Graf, dilakukan pembuatan struktur Graf pada program dengan penyusunan grafnya terdiri dari titik koordinat jalan dan persimpangan yang ada pada lokasi penelitian untuk. Struktur Graf ini dibuat sebagai representasi jalan yang digunakan untuk membuat rute alternatif.

Pada penelitian ini, LBS diterapkan untuk membuat informasi penutupan jalan dan membuat fitur notifikasi peringatan penutupan jalan. Untuk membuat informasi penutupan jalan, salah satu data yang sematkan dalam informasi tersebut adalah data koordinat lokasi penutupan jalan. Data lokasi tersebut didapatkan dengan membuat tampilan *Map View* pada halaman tambah informasi. Melalui *Map View* tersebut, User Public sebagai pengguna aplikasi *mobile* dapat menentukan titik penutupan jalan. Sehingga saat informasi di-*publish* oleh *operator*, informasi penutupan jalan yang dilengkapi dengan informasi titik penutupan jalan dapat diakses. Informasi dapat dilihat melalui tampilan *Maps*.

Kemudian untuk membuat notifikasi peringatan penutupan jalan, peneliti menerapkan *package* "com.google.android.gms.location". Pada *package* tersebut terdapat beberapa *class* yang berhubungan dengan *Geofence*. Cara kerja *Geofence* adalah dengan memberikan radius pada suatu titik koordinat pada *Map*. Pada penelitian ini, peneliti mengatur radius sejauh 100 meter. Terdapat 3 jenis transisi *Geofence*, yaitu *enter, dwell* dan *exit*. Transisi *enter* adalah transisi dengan kondisi perangkat memasuki daerah radius. Transisi *dwell* adalah transisi dengan kondisi perangkat menjelajah di dalam radius. Transisi *exit* adalah transisi dengan kondisi perangkat keluar dari daerah radius. Peneliti hanya menerapkan transisi *enter* pada aplikasi, karena yang ingin dibuat adalah fungsi untuk menerima

notifikasi peringatan ketekita *User Public* memasukan radius penutupan jalan.

Penerapan struktur Graf dilakukan dengan membuat tabel persimpangan pada *database*. Tabel ini berisi data titik jalan dan persimpangan. Atribut data ini antara lan idPersimpangan, label, *latlng* dan *node* tetangga. Representasi grap dibuat dengan menadikan setiap data pada tabel tersebut memiliki node tetangga. Sehingga pada diimplementasikan pada program, data dari tabel persimpangan akan membentuk representasi jalan yang selanjutnya digunakan untuk membuat rute. dalam sebuah rute, terdapat titik asal dan titik tujuan, serta pada penelitian ini terdapat titik penutupan jalan yang nantinya akan diolah sebagai *node* hambatan yang diabaikan dalam pembuatan rutenya. Ketiga jenis node tersebut (titik asal, titik tujuan dan titik hambatan) akan dihubungkan dengan struktur Graf yang sudah sebelumnya, yaitu yang tersusun dari data-data tabel persimpangan. Tujuannya adalah agar dapat menelusuri dan mengenali titik asal, titik tujuan dan titik penutupan jalan melalui struktur Graf. Untuk rutenya dihasilkan dengan menggunakan *rules* pencarian rute yang dibuat dengan memodifikasi algoritma *Breadth First Search*.

5.2. SARAN

Dari hasil penelitian yang didapatkan, aplikasi yang dibuat memiliki beberapa kekurangan. Yang pertama adalah mengenai struktur Graf yang diterapkan dalam sistem beserta *rules*-nya. Pada penelitian ini, dimungkinkan adanya penambahan titik jalan baru secara permanen, yaitu dengan menyimpan titik jalan baru tersebut ke dalam *database*. Namun *rules* yang diterapkan kurang optimal, karena titik jalan yang baru hanya akan memiliki 1 titik jalan tetangga (node tetangga) dari titik jalan yang ada sebelumnya. Sedangkan yang lebih baik adalah penambahan titik jalan juga dapat dilakukan diantara 2 titik jalan yang sudah ada, dengan menjadikan komposisi node tetangga berubah, baik pada titik jalan yang baru ditambahkan, ataupun titik jalan yang sudah ada sebelumnya.

Mengenai notifikasi peringatan penutupan jalan, aplikasi yang dibuat pada penelitian ini bekerja kurang baik jika berada pada kondisi *force close*. Ada baiknya jika pada saat aplikasi ditutup atau *force close*, aplikasi harus tetap

melakukan reqeust *current location* atau *update* lokasi. Melihat dari pengujian yang dilakukan terhadap notifikasi peringatan penutupan jalan, *Geofence* akan bekerja lebih akurat dan aktual jika pada perangkat tersebut terdapat *service update current location* yang berjalan pada latar belakang.

DAFTAR PUSTAKA

A.S., Rosa, dan M. Shalahudin. 2016. *Rekayasa Perangkat Lunak - Terstruktur dan Berorientasi Objek*. Bandung:Informatika.

Amit Kushwaha, Vineet Kushwaha. 2011. *Location Based Services using Android Mobile Operating System International Journal of Advances in Engineering & Technology*. 1(1), 14-20

B. A. Candra, K. Muludi dan A.R. Irawati. 2012. *Rancang Bangun Sistem Informasi Manajemen Terpadu (SIMANTEP) Online PT. PLN (Persero) Sektor Pembangkitan Tarahan Lampung Dengan Metode Extreme Programming*. Jurnal Komputasi, 1(1).

Budi Prasetiyo dan Maulidia Rahmah Hidayah. 2014. *Penggunaan Metode Depth First Search (DFS) dan Breadth First Search (BFS) pada Strategi Game Kamen Rider Decade Versi 0.3*. *Scientific Journal of Informatics*. 1(2). 161-167.

Edy Budiman. 2016. *Pemanfaatan Teknologi Location Based Service Dalam Pengembangan Aplikasi Profil Kampus Universitas Mulawarman Berbasis Mobile*. Ilmiah Ilkom, 8(3).

Oktaviani, N., & Hutrianto, H. (2016). Extreme Programming sebagai Metode Pengembangan E-keuangan pada Pondok Pesantren Qodratullah. Jurnal Ilmiah MATRIK, 18(2), 163-178.

Mira Kusmira dan Taufiqurrochman. 2017. *Pemanfaatan Aplikasi Graf Pada Pembuatan Jalur Angkot 05 Tasikmalaya*. Semnastek.

Muhammad Iqnaul Haq dan Taufik Hery Purwanto. 2014. *Pembuatan Rute Alternatif Berbasis Web-Gis Untuk Menghindari Kemacetan Lalulintas Di Kota Tangerang Selatan*. Bumi Indonesia, 3(2).

Nur Alam dan Mukhlis Amin. 2015. *Aplikasi Pemilihan Rute Alternatif Akibat Kemacetan Lalu Lintas di Kota Makassar Menggunakan Google API dan ASP.Net*. Jurnal Pekommas, 18(2), 93-104.

Raisya Rahmi, Risa perdana Sari dan Rahmat Suhatman. 2016. *Pendekatan Metodologi Extreme Programming pada Aplikasi E-Commerce (Studi kasus Sistem Informasi Penjualan Alat-alat Telekomunikasi)*. Komputer Terapan, 2(2), 83-92.

Rusdiana, L., & Marfuah, M. 2017. *The Application of Determining Students' Graduation Status of STMIK Palangkaraya Using K-Nearest Neighbors Method*. International Conference on Environment and Technology (IC-Tech). Pekanbaru, Indonesia: IOP Conf. Series: Earth and Environmental Science.

Viktor Handrianus Pranatawijaya, Deddy Ronaldo, dan Farhani. 2018. *Penerapan Location Based Service Pada Jurusan Teknik Informatika Fakultas Teknik Universitas Palangka Raya*. Jurnal Teknologi Informasi, 12(1), 65-73.