RANCANG BANGUN APLIKASI PENCARIAN HALTE TERDEKAT BUS TRANS SARBAGITA MENGGUNAKAN ALGORITMA FLOYD WARSHALL BERBASIS ANDROID

SKRIPSI

Oleh

Anisia Karnia Septi

NIM 112410101084

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2016

RANCANG BANGUN APLIKASI PENCARIAN HALTE TERDEKAT BUS TRANS SARBAGITA MENGGUNAKAN ALGORITMA FLOYD WARSHALL BERBASIS ANDROID

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan pendidikan di Program Studi Sistem Informasi Universitas Jember dan mendapat gelar Sarjana Sistem Informasi

Oleh

Anisia Karnia Septi

NIM 112410101084

PROGRAM STUDI SISTEM INFORMASI

UNIVERSITAS JEMBER

2016

PERSEMBAHAN

Skripsi ini saya persembahkan untuk :

1. Puji syukur kepada Allah SWT yang Maha Pengasih lagi Maha Penyayang, Maha Pelindung, Maha Pemberi Semangat dan Kekuatan. Sholawat serta salam selalu terlimpahkan untuk Rosulullah Muhammad SAW
2. Ibunda terkasih Rina Agustini, SH dan Ayahanda Ir. Agus Noerdjembana terima kasih telah berkorban sedemikian besar untuk anakmu yang belum bisa membalas dan tidak akan pernah bisa membalas sepadan dengan apa yang telah engkau berikan, keikhlasan, ketulusan, kasih sayang, kerja keras, bimbingan, ajaran dalam hidup dan beragama
3. Saudara satu-satunya Rachmad Dwiki Permana Putra yang sangat aku sayangi.
4. Seseorang yang kelak akan menjadi suamiku.
5. Sahabatku Ayusep, Ayunov, Anif, Desi Wulan, Dwi, Devi P, Yunita, Iin, Qilba, Hida, Ulil, Yuni, alayers, lebayers dan teman-teman yang tidak cukup bila disebut namanya.
6. Guru-guruku sejak taman kanak-kanak sampai dengan perguruan tinggi.
7. Almamater Program Studi Sistem Informasi Universitas Jember.

MOTTO

"Sesuatu yang belum dikerjakan, seringkali tampak mustahil; kita baru yakin kalau kita telah berhasil melakukannya dengan baik".

-Evelyn Underhill-

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Anisia Karnia Septi

NIM : 112410101084

menyatakan sesungguhnya bahwa karya ilmiah yang berjudul "Rancang Bangun Aplikasi Pencarian Halte Terdekat Bus Trans SARBAGITA Menggunakan Algoritma Floyd Warshall Berbasis Android" adalah benar-benar hasil karya sendiri, kecuali kutipan yang sudah saya sebutkan sumbernya, belum pernah diajukan pada institusi mana pun dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika ternyata di kemudian hari pernyataan ini tidak benar.

Jember, 02 Juni 2016

Yang menyatakan,

Anisia Karnia Septi

NIM. 112410101084

SKRIPSI

RANCANG BANGUN APLIKASI PENCARIAN HALTE TERDEKAT BUS TRANS SARBAGITA MENGGUNAKAN ALGORITMA FLOYD WARSHALL BERBASIS ANDROID

Oleh:

Anisia Karnia Septi

NIM 112410101084

Pembimbing :

Dosen Pembimbing Utama	: Dr. Saiful Bukhori, ST., M.Kom
	NIP. 196811131994121001
Dosen Pembimbing Pendamping	: Windi Eka Yulia Retnani, S.Kom., MT
	NIP. 198403052010122002

PENGESAHAN PEMBIMBING

Skripsi berjudul "Rancang Bangun Aplikasi Pencarian Halte Terdekat *Bus* Trans SARBAGITA Menggunakan Algoritma Floyd Warshall Berbasis Android", telah diuji dan disahkan pada:

hari, tanggal : Kamis, 2 Juni 2016

tempat : Program Studi Sistem Informasi Universitas Jember

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Dr. Saiful Bukhori, ST., M.Kom	Windi Eka Yulia Retnani, S.Kom., MT
NIP. 196811131994121001	NIP. 198403052010122002

PENGESAHAN PENGUJI

Skripsi berjudul "Rancang Bangun Aplikasi Pencarian Halte Terdekat *Bus* Trans SARBAGITA Menggunakan Algoritma Floyd Warshall Berbasis Android", telah diuji dan disahkan pada:

hari, tanggal : 2 Juni 2016

tempat : Program Studi Sistem Informasi Universitas Jember

Penguji I,	Penguji II,
Drs.Antonius Cahya P, M.App.,Sc., Ph.D	Yanuar Nurdiansyah, ST., M.Cs
NIP. 196909281993021001	NIP. 198201012010121004

Mengesahkan

Ketua Program Studi,

Prof. Drs. Slamin, M.Comp.Sc., Ph.D

NIP. 196704201992011001

ABSTRAK

Pengguna *bus* Trans SARBAGITA sering sekali mengalami kesusahan dalam mencari informasi tentang *bus* ini, khususnya dimana letak halte dan rute apa saja yang tersedia. Pencarian lokasi halte membutuhkan waktu yang lama jika mencari secara manual terlebih tidak familiar dengan wilayah Bali. Selain itu, menuju halte yang tidak sesuai dengan tujuan karena kurangnya informasi tentang rute yang dilalui. *Smartphone* sudah banyak digunakan masyarakat saat ini terutama *platform* android. Aplikasi Trans SARBAGITA *bus stop finder* ini berbasis android dan dapat digunakan dalam keadaan apapun selama tersambung dengan jaringan internet. Proses untuk mendapatkan posisi pengguna pada aplikasi ini adalah dengan menggunakan fitur *Global Positioning System* (GPS) pada perangkat *smartphone* dan *Google Maps API* sebagai pengembangnya. Perhitungan untuk mendapatkan lokasi halte terdekat dari posisi menggunakan algoritma *Floyd Warshall.* Perhitungan menggunakan parameter jarak yang didapat dari titik koordinat posisi pengguna dan titik koordinat halte. Hasil dari aplikasi ini berupa halte terdekat dari posisi pengguna sesuai rute pilihan dan mendapatkan informasi lintasan menuju lokasi halte tersebut. Algoritma ini menghitung bobot terkecil dari semua jalur yang menghubungkan sebuah pasangan titik dalam graf berarah dan berbobot, dan melakukannya sekaligus untuk semua pasangan titik sehingga hasil halte terdekat yang didapat lebih optimal.

Kata kunci : : Android*, Floyd Warshall, Halte,* Rute Terpendek

RINGKASAN

Rancang Bangun Aplikasi Pencarian Halte Terdekat *Bus* Trans SARBAGITA Menggunakan Algoritma Floyd Warshall Berbasis Android; Anisia Karnia Septi, 112410101084; 2016, 135 HALAMAN; Program Studi Sistem Informasi Universitas Jember.

Bali merupakan salah satu pusat pariwisata utama di Indonesia yang terkenal sampai ke pelosok dunia. Peningkatan jumlah wisatawan dan perkembangan pariwisata di bali terbukti dapat memacu pertumbuhan ekonomi masyarakat. Pertumbuhan ekonomi masyarakat berdampak pada peningkatan jumlah kepemilikan kendaraan pribadi yang menyebabkan masalah kemacetan di Bali. Hingga saat ini kemacetan menjadi salah satu masalah utama di Bali, khususnya di kawasan Denpasar, Badung, Gianyar, dan Tabanan. Pemerintah Provinsi Bali membuat kebijakan untuk masalah kemacetan agar segera teratasi, yaitu dengan menyediakan transportasi publik bernama *bus* Trans SARBAGITA (Dennpasar, Badung, Gianyar, Tabanan). Konsep *bus* Trans SARBAGITA hampir sama dengan *Busway* Jakarta, hanya saja tidak mempunyai jalur khusus. Trans SARBAGITA resmi beroperasi sejak bulan Agustus 2011, hingga saat ini mempunyai 2 koridor utama dengan rute Kota – GWK PP dan Batu Bulan – Nusa Dua PP. *Bus* Trans SARBAGITA tidak hanya digunakan oleh masyarakat Bali, namun juga untuk para wisatawan yang datang.

Sejak awal diresmikan hingga saat ini, respon dari masyarakat maupun wisatawan masih kurang karena banyak dari mereka yang tidak tahu mengenai informasi tentang *bus* Trans SARBAGITA seperti rute apa saja yang dilalui, letak halte *bus,* harga tiket, dan waktu beroperasinya. Pengguna *bus* Trans SARBAGITA sering mengalami kesulitan dalam mencari letak halte yang sesuai dengan tujuan mereka, karena itu diperlukan aplikasi yang dapat memberikan informasi halte terdekat dari posisi pengguna. Tentunya aplikasi yang akan dibangun adalah aplikasi berbasis android, mengingat *smartphone* sudah banyak digunakan oleh masyarakat. Android

menjadi salah satu *platform* yang banyak digemari karena berbagai aplikasi yang memudahkan perkerjaan pengguna *smartphone* dengan berbagai fitur salah satunya dalam mencari informasi dan menuju suatu tempat tertentu, contohnya pencarian halte *bus*.

Pencarian halte *bus* terdekat dapat menerapkan metode pencarian rute terdekat. Salah satu metode yang bisa digunakan untuk menyelesaikan suatu masalah dalam proses pencarian rute terpendek yaitu dengan menggunakan algoritma *Floyd Warshall.* Metode *Floyd Warshall* merupakan salah satu algoritma pencarian yang dapat digunakan dalam menghitung rute terpendek dan membandingkan semua kemungkinan rute pada graf untuk semua simpul yang ada. Sistem pencarian halte terdekat *bus* Trans SARBAGITA ini berbasis android dimana aplikasi dapat digunakan dimanapun dan kapanpun. Pencarian halte terdekat *bus* Trans SARBAGITA menggunakan jarak sebagai parameter dalam perhitungan menggunakan algoritma *Floyd Warshall*. Hasil pencarian halte terdekat *bus* Trans SARBAGITA ini berdasarkan pada rute pilihan *user* yang ada pada koridor 1 atau 2.

Aplikasi Trans SARBAGITA *Bus Stop Finder* dapat diterapkan pada android versi 4.0 keatas. Beberapa fitur aplikasi Trans SARBAGITA *Bus Stop Finder* adalah *My Locatian* dan *Bus Route*. Fitur *My Location* merupakan fitur untuk mendapatkan posisi *user* saat menggunakan aplikasi. Fitur *Bus Route* merupakan fitur untuk mendapatkan halte terdekat beserta lintasan menuju halte berdasarkan rute yang dipilih *user* dalam koridor 1 atau 2.

PRAKATA

Puji syukur ke hadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul "Rancang Bangun Aplikasi Pencarian Halte Terdekat *Bus* Trans SARBAGITA Menggunakan Algoritma Floyd Warshall Berbasis Android". Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Drs. Slamin, M.CompSc., Ph.D., selaku Ketua Program Studi Sistem Informasi Universitas Jember.
2. Dr. Saiful Bukhori ST.,M.Kom selaku Dosen Pembimbing Utama dan Windi Eka Yulia Retnani S.Kom.,MT selaku Dosen Pembimbing Anggota yang telah petunjuk, bimbingan, koreksi, serta saran hingga skripsi ini dapat tersusun dengan baik
3. Dr. Saiful Bukhori, ST., M.Kom, selaku Dosen Pembimbing Akademik yang telah membimbing selama penulis menjadi mahasiswa.
4. Seluruh Bapak dan Ibu dosen beserta staf karyawan di Program Studi Sistem Informasi Universitas Jember.
5. Ayahanda Ir. Agus Noerdjembana dan Ibunda Rina Agustini, SH yang telah sangat memotivasi penulis.
6. Adikku Rachmad Dwiki Permana Putra yang selalu memberi semangat.
7. Keluarga besar Program Studi Sistem Informasi, Dosen, Staf, HIMASIF, sahabat-sahabat terbaik Desi Wulan, Ayusep, Ayunov, Anif, Dwi, Zia, Uni,

Tari, Hida, Ulil, Yuni, kelompok Alayers, lebayers dan lainnya yang telah membantu selama menimba ilmu disini.

8. Keluarga Besar Himpunan Mahasiswa Sistem Informasi (HIMASIF) periode 2011-2012 dan UKM Kesenian.
9. Teman-teman mahasiswa Program Studi Sistem Informasi Universitas Jember terkhusus NEFOTION.
10. Semua pihak yang telah membantu baik tenaga maupun pikiran dalam pelaksanaan kegiatan penelitian dan penyusunan skripsi ini.

Dengan harapan bahwa penelitian ini nantinya akan terus berlanjut dan berkembang kelak, penulis juga menerima segala kritik dan saran dari semua pihak demi kesempurnaan skripsi ini. Akhirnya penulis berharap, semoga skripsi ini dapat bermanfaat.

Jember, 17 Juni 2016

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1. PENDAHULUAN

Bab ini merupakan bab awal dari tugas akhir yang didalamnya terdapat latar belakang, rumusan masalah, tujuan, manfaat, dan sistematika penulisan.

1.1 Latar Belakang

Bali merupakan salah satu pusat pariwisata utama di Indonesia yang telah terkenal sampai ke pelosok dunia. Tersedianya berbagai objek wisata alam maupun budaya mengakibatkan jumlah wisatawan meningkat setiap tahun. Peningkatan jumlah wisatawan merupakan salah satu indikator perkembangan pariwisata di Provinsi Bali dan terbukti dapat memacu pertumbuhan perekonomian masyarakat.

Pertumbuhan ekonomi masyarakat yang terus meningkat akan berdampak pada peningkatan jumlah kepemilikan kendaraan pribadi setiap tahun. Meningkatnya volume kendaraan yang tidak seimbang dengan volume jalan menyebabkan kemacetan lalu lintas. Kemacetan menjadi salah satu masalah utama di Bali, khususnya di kawasan Denpasar, Badung, Gianyar dan Tabanan yang merupakan pusat pemerintahan, niaga, perhotelan, serta wisata alam maupun budaya. Pemerintah Provinsi Bali tentunya harus memberikan solusi agar masalah kemacetan segera teratasi.

Salah satu kebijakan yang dibuat oleh pemerintah Bali untuk mengatasi masalah kemacetan adalah menyediakan transportasi publik bernama *Bus* Trans SARBAGITA (Denpasar, Badung, Gianyar, Tabanan). Konsep *bus* Trans SARBAGITA hampir sama dengan *Busway* Trans Jakarta yang merupakan sistem *bus* cepat atau *mass rapid transit* yang dilengkapi dengan fasilitas halte dan rute yang jelas. Hanya saja *bus* Trans SARBAGITA tidak mempunyai jalur khusus seperti *busway* Trans Jakarta. *Bus* ini tidak hanya digunakan untuk masyarakat Bali saja, namun juga untuk para wisatawan yang datang sebagai pilihan transportasi murah untuk mengunjungi objek wisata.

Sudah berjalan hampir 5 tahun *bus* Trans SARBAGITA beroperasi dengan 2 koridor yang sudah tersedia, namun respon dari masyarakat maupun wisatawan masih kurang karena masih sedikit informasi mengenai *bus* tersebut. Banyak pengguna *bus* yang masih tidak tahu adanya *bus* Trans SARBAGITA, rute mana saja yang dilalui *bus*, dimana letak halte *bus*, harga tiket dan waktu beroperasinya. Pengguna sering mengalami kesulitan dalam menemukan letak halte dan mencari halte mana yang harus dituju agar mendapatkan *bus* Trans SARBAGITA yang sesuai dengan tujuan mereka. Kesulitan masyarakat mencari halte, contohnya seseorang ingin mencari halte terdekat *bus* Trans SARBAGITA untuk menuju ke daerah Sanur dengan posisi saat ini di daerah Denpasar, tetapi pengguna malah menuju halte Kamboja untuk menaiki *bus*, padahal rute *bus* dari halte tersebut tidak melewati daerah tujuan, karena itu diperlukan aplikasi yang dapat memberikan informasi halte terdekat dari posisi pengguna. Tentunya aplikasi yang akan dibangun tersebut adalah aplikasi yang mudah diakses dimana saja dan kapan saja oleh masyarakat. Aplikasi yang tepat adalah aplikasi yang berbasis android, mengingat *Smartphone* sekarang ini sudah banyak digunakan oleh masyarakat.

Para pengguna *smartphone* berbasis android berasal dari berbagai kalangan dan umur. *Smartphone* tidak hanya digunakan untuk alat komunikasi, bermain *game,* atau keperluan sosial media saja, namun juga dimanfaatkan untuk mencari informasi menuju suatu tempat tertentu dengan mudah. Pencarian informasi untuk menuju suatu tempat, contohnya pencarian halte *bus* yang banyak menjadi tujuan para pengguna transportasi umum untuk menggunakan jasa transportasi *bus*.

Pencarian halte *bus* terdekat dapat menerapkan metode pencarian rute terdekat yang banyak diterapkan pada berbagai bidang untuk mengoptimasi kinerja suatu sistem baik untuk meminimalkan biaya ataupun mempercepat jalannya suatu proses. Pencarian suatu lokasi secara *manual* dapat membutuhkan waktu yang lama. Salah satu metode yang bisa digunakan untuk menyelesaikan suatu masalah dalam proses pencarian rute terpendek yaitu dengan menggunakan algoritma *Floyd Warshall.* Metode *Floyd Warshall* merupakan salah satu algoritma pencarian yang dapat

digunakan dalam menghitung rute terpendek dan membandingkan semua kemungkinan rute pada graf untuk semua simpul yang ada.

Penelitian yang berjudul "Penentuan Jarak Terpendek Rute Transmusi dengan Algoritma *Floyd-Warshall*" menerapkan algoritma *Floyd-Warshall* untuk mencari rute terpendek *Bus* Transmusi. Penelitian ini bertujuan untuk penggunaan waktu agar menjadi lebih efekfif dan permasalahan penumpang atau calon penumpang dapat diselesaikan (Kriswanto R.Y., Bendi R.K.J, 2014).

Berdasarkan latar belakang tersebut, maka dibutuhkan suatu penerapan aplikasi berbasis android yang mampu membantu calon penumpang *bus* menemukan halte terdekat *bus* Trans SARBAGITA yang diterapkan menggunakan algoritma *Floyd Warshall*.

1.2 Rumusan Masalah

Permasalahan yang telah diuraikan diatas, maka rumusan masalah dalam penelitian ini adalah sebagai berikut :

1. Bagaimana membuat aplikasi yang dapat memberikan informasi halte *bus* Trans SARBAGITA terdekat pada *platform* android?
2. Bagaimana menerapkan algoritma *Floyd Warshall* pada aplikasi pencarian halte terdekat *bus* Trans SARBAGITA ?

1.3 Tujuan dan Manfaat

Berikut merupakan tujuan yang ingin dicapai dan manfaat yang ingin didapat dalam penelitian ini.

1.3.1 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah :

1. Membuat aplikasi yang dapat memberikan informasi halte *bus* Trans SARBAGITA terdekat pada *platform* android.

2. Membuat aplikasi pencarian halte terdekat *bus* Trans SARBAGITA dengan menerapkan algoritma *Floyd Warshall*.

1.3.2 Manfaat

Manfaat yang ingin didapatkan dari penelitian ini adalah :

a. Manfaat Bagi Akademis

Hasil penelitian ini diharapkan dapat memberikan kontribusi dan masukan bagi siapa saja yang membutuhkan informasi yang berhubungan dengan judul penelitian ini. Selain itu, hasil penelitian ini merupakan suatu upaya untuk menambah varian judul penelitian yang ada di Program Studi Sistem Informasi Universitas Jember.

b. Manfaat Bagi Peneliti

1. Mengetahui bagaimana proses penerapan algoritma *Floyd Warshall* untuk aplikasi pencarian halte terdekat *bus* Trans SARBAGITA berbasis android.
2. Sebagai media penyelesaian Tugas Akhir untuk jenjang S1 pada Program Studi Sistem Informasi Universitas Jember

c. Manfaat bagi objek penelitian

Memberikan inovasi baru kepada instansi tempat penelitian dilakukan mengenai penggunaan aplikasi pencarian halte terdekat *Bus* Trans SARBAGITA.

1.4 Batasan Masalah

Batasan masalah dalam penelitian ini adalah :

1. Alur operasional *bus* disesuaikan dengan sistematika yang berlaku di UPT Trans SARBAGITA.
2. Aplikasi ini menggunakan algoritma *Floyd Warshall* untuk mencari halte terdekat dan menampilkan rute menuju halte dari posisi *user*.
3. Aplikasi ini mencakup koridor 1 dan koridor 2 *Bus* Trans SARBAGITA. Pada setiap koridor tersedia beberapa rute pilihan.

4. Pencarian halte terdekat berdasarkan pada koridor dan rute *Bus* Trans SARBAGITA yang diinginkan *user.*
5. Hanya jarak yang menentukan dalam pencarian halte terdekat.
6. Hambatan seperti lampu merah, kerusakan dan kemacetan jalan diabaikan.
7. Tidak dibahas waktu dan kecepatan transportasi dari posisi *user* menuju halte terdekat.
8. Aplikasi ini hanya dapat di jalankan pada *platform* android 4.0 ke atas.

1.5 Sistematika Penulisan

Sistematika penulisan dalam penyusunan tugas akhir ini adalah sebagai berikut :

a. Pendahuluan

Bab ini berisi latar belakang, perumusan masalah, tujuan dan manfaat, metodologi penelitian dan sistematika penulisan skripsi.

b. Tinjauan Pustaka

Bab ini berisi tentang kajian pustaka, penelitian terdahulu dan informasi apa saja yang digunakan dalam penelitian ini. Dimulai dari memaparkan penelitian dahulu sampai kajian pustaka mengenai penelitian ini.

c. Metodologi Penelitian

Bab ini menguraikan tentang metode apa yang dilakukan selama penelitian. Dimulai dari tahap pencarian permasalahan hingga pengujian aplikasi pencarian halte terdekat *bus* Trans SARBAGITA akan dibuat.

d. Hasil dan Pembahasan

Bab ini menjelaskan tentang hasil dan pembahasan dari penelitian yang telah dilakukan. Dengan memaparkan hasil penelitian dan hasil percobaan implementasi sistem.

e. Penutup

Bab ini berisi kesimpulan dari penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Bab ini memaparkan teori – teori dan pustaka yang digunakan dalam penelitian. Teori – teori ini diambil dari buku literatur dan jurnal. Berikut merupakan teori – teori yang digunakan dan dibahas dalam penelitian ini :

2.1 Penelitian Terdahulu

Adapun penelitian terdahulu sehingga penelitian ini muncul adalah sebagai berikut :

1. Penelitian yang berjudul "Aplikasi *Mobile* Pencarian Rute Terpendek Lokasi Fasilitas Umum Berbasis Android Menggunakan Algoritma *Floyd Warshall"* dilakukan oleh Rizky Yusaputra dari Universitas Islam Negeri Sultan Syarif Kasim Riau (2013). Pada penelitian ini dilakukan perancangan aplikasi *mobile* berbasis android yang berfungsi untuk mencari lokasi fasilitas umum dengan rute terpendek menggunakan algoritma *Floyd Warshall.* Sehingga pengguna dapat melihat posisi saat itu dan posisi beberapa fasilitas umum yang ada, juga dapat mengetahui rute mana yang harus mereka lewati untuk mencapai lokasi masing-masing fasilitas umum yang ada di kota Pekanbaru dengan cara yang lebih mudah dan cepat. Pengujian yang dilakukan peneliti meliputi pengujian dari aspek jarak dan waktu yang dapat ditempuh. Hasil yang diinginkan peneliti dari sistem pencarian rute terpendek adalah *direction* dengan rute terpendek, total jarak dan waktu yang ditempuh. Hasil dari pengujian dalam melakukan pencarian rute terpendek di 15 lokasi tujuan yang berbeda dapat disimpulkan bahwa algoritma *Floyd Warshall* ini sudah dapat menghasilkan solusi yang lebih optimum daripada algoritma pencarian rute terpendek yang lainnya, akan tetapi untuk waktu yang dibutuhkan dalam melakukan proses pencarian tersebut relatif lebih lama dari algoritma yang lain (Yusaputra, 2013).

2. Penelitian lain dengan Rancang Bangun Aplikasi untuk Menentukan Jalur Terpendek Menggunakan Algoritma *Floyd Warshall* di Lokasi Wisata Purbalingga dilakukan oleh Irfan Ardiansyah dan Dimara Kusuma Hakim Universitas Muhammadyah Purwokerto (2012). Pada penelitian ini, diteliti bagaimana wisatawan atau masyarakat dapat dengan mudah mencari jalur terpendek untuk menuju lokasi wisata di Purbalingga. Hasil dari penelitian ini disebutkan bahwa aplikasi ini sudah dapat memudahkan wisatawan ataupun masyarakat dalam menentukan jalur terpendek menuju lokasi wisata di Purbalingga dibantu dengan adanya navigasi jalur terpendek membuat aplikasi ini lebih informatif. Penelitan ini menggunakan algoritma *Floyd Warshall* untuk menentukan jalur terpendek dengan titik awal di alun-alun dan tujuan di berbagai lokasi wisata Purbalingga. Titik awal di alun-alun yang sudah ditentukan ini kurang efisien dalam penggunaan aplikasi (Ardiansyah I., Hakim D.K, 2012).

2.2 Angkutan Umum

Angkutan umum pada dasarnya merupakan sarana untuk memindahkan orang dan barang dan suatu tempat ke tempat lain. Tujuannya untuk membantu orang atau kelompok orang dalam menjangkau tempat yang dikehendaki, atau mengirim barang dan tempat asal ke tempat tujuan. Manfaat pengangkutan dapat dilihat dan berbagai kehidupan masyarakat yang dapat dikelompokkan menjadi tiga bagian yaitu manfaat ekonomi, soaial dan politik.

Terdapat 2 (dua) sistem pemakai angkutan umum berdasarkan peraturan Direktorat jenderal Perhubungan Darat tahun 1994, yaltu sebagai berikut:

a. Sistem sewa, yaitu kendaraan yang bisa dioperasikan balk oleh operator maupun oleh penyewa. Dalam hal ini tidak ada rute dan jadwal tertentu yang hams diikuti oleh pemakal. Sistem ini sering disebut sebagal *demand responsive system,* karena penggunaannya yang tergantung pada adanya permintaan. Contoh jenis ini adalah angkutan jenis taksi.

b. Sistem penggunaan bersama, yaitu kendaraan dioperasikan oleh operator dengan rute dan jadwal yang tetap. Sistem ini dikenal dengan *transit system*. Terdapat dua jenis transit, yaitu sebagai berikut:

1. Para transit, yaitu tidak ada jadwal yang pasti dan kendaraan dapat berhenti untuk menaikkan dan menurunkan penumpang di sepanjang rutenya. Contohnya adalah angkutan kota atau angkutan pedesaan.
2. Mass transit, yaitu jadwal dan tempat hentinya Iebih pasti dan teratur. Contohnya adalah kereta api (Wiwaha, 2013).

2.3 Trans SARBAGITA

Layanan Trans SARBAGITA diluncurkan oleh Bapak Gubernur Bali Mangku Pastika pada tanggal 17 Agustus 2011 dan mulai dioperasikan memberi layanan pada tanggal 18 Agustus 2011. Dasar penyediaan angkutan umum massal Trans SARBAGITA yaitu berdasarkan pada UU Nomor 22 tahun 2009 tentang Lalu Lintas dan Angkutan Jalan, selain itu terdapat Peraturan Daerah 16 tahun 2009 tentang Rencana Tata Ruang Wilayah (RTRW) Provinsi Bali dan Peraturan Daerah tahun 2009 tentang Rencana Pembangunan Jangka Panjang Daerah (RPJPD) Provinsi Bali tahun 2005-2025).

Adapun tujuan dari penyediaan angkutan umum massal Trans SARBAGITA ini adalah untuk menata/restrukturasi jaringan trayek angkutan umum yang ada kedalam satu kesatuan sistem jaringan trayek yang terkoneksi, terintegrasi dan terpadu, sebagai satu kesatuan sistem jaringan pelayanan untuk memberikan pilihan perjalanan bagi masyarakat. Sedangkan sasarannya adalah tersedianya pilihan pergerakan orang dari pusat produksi ke pusat distribusi secara efisisen, lancar, aman, dan nyaman dalam rangka meningkatkan produktivitas jaringan jalan dan untuk harapannya adalah terwujudnya kelancaran dan ketertiban lalu lintas di jalan untuk menunjang kegiatan perekonomian daerah.

Disamping itu, keberadaan Trans SARBAGITA juga diharapkan dapat mengurangi tingkat kemacetan lalu lintas dan menekan penggunaan kendaraan pribadi guna melestarikan lingkungan dari ancaman polusi udara. Oleh karena itu, Trans SARBAGITA juga mengampanyekan penghijauan melalui slogan *Go Green* yang tertulis di *bus* Trans SARBAGITA. Implementasi angkutan umum Trans SARBAGITA ditunjukkan dengan konsep menciptakan efisiensi penggunaan ruang jalan dan menjangkau seluruh kawasan melalui :

a. Restrukturisasi jaringan trayek Kota Denpasar dan wilayah sekitarnya ke dalam satu kesatuan sistem jaringan pelayanan angkutan umum lintas dan dalam kota / kabupaten di wilayah Denpasar, Badung, Gianyar dan Tabanan (SARBAGITA), meliputi 17 trayek utama dan 36 trayek cabang / ranting atau yang sering disebut *feeder*. Trayek feeder untuk pertama kalinya diluncurkan pada bulan September 2012 di Kota Denpasar, dan keberadaan trayek pengumpan ini sangat membantu masyarakat Kota Denpasar menjangkau halte-halte *bus* Trans SARBAGITA di daerah Kota Denpasar baik pada koridor I maupun II. Terutama bagi mereka yang melakukan perjalanan cukup jauh dan tidak memiliki kendaraan pribadi.
b. Penggunaan kendaraan dengan kapasitas sesuai dengan panjang perjalanan / struktur trayek yaitu:
    1. *Bus* : untuk trayek utama
    2. Elf / *minibus* : untuk trayek cabang
    3. *Microlet* : untuk trayek ranting
c. Beroperasi setiap hari dan terjadwal (penumpang menunggu *bus*, bukan *bus* menunggu penumpang).
d. Berhenti (menaikkan dan menurunkan penumpang) hanya pada halte dan *bus stop* yang ditetapkan
e. Tarif terjangkau
f. Menerapkan sistem pembelian layanan (*by the service*)

Standar Pelayanan Minimal (SPM) *bus* Trans SARBAGITA meliputi kenyamanan, keamanan, terjadwal, terjangkau, dengan rincian antara lain:

a. Trayek utama dilayani dengan *bus* sedang kapasitas 30 orang (20 orang duduk ditambah 10 orang berdiri).
b. Trayek cabang dilayani kendaraan Elf dengan kapasitas 12 orang.
c. Awak kendaraan terdiri dari Pramudi (Sopir) dan Pramujasa.
d. Halte dengan sistem terbuka, dengan pertimbangan kebutuhan lahan lebih kecil, bila dibandingkan dengan sistem tertutup.
e. Ketinggian lantai halte 80 cm untuk *bus* sedang dan 110 cm untuk *bus* besar.
f. Kendaraan hanya diijinkan menaikkan dan menurunkan penumpang di halte-halte yang telah ditetapkan.
g. Waktu pengoperasian *bus* dilakukan setiap hari dengan asal sampai tujuan melalui rute tetap dari jam 05.00 – 21.00 WITA.
h. Maksimum kecepatan *bus* dalam kota 40 km/jam dan luar kota 50 km/jam (kecepatan rata-rata 20 km/jam).
i. Waktu menaikkan dan menurunkan penumpang di setiap halte persinggahan selama 60 detik dengan toleransi 30 detik.

Penetapan tarif penumpang angkutan umum Trans SARBAGITA didasarkan pada SK Gubernur Bali Nomor 11 Tahun 2011 tanggal 11 April dimana untuk penumpang umum dikenai tarif sebesar Rp 3.500 dan untuk pelajar / mahasiswa sebesar Rp 2.500 (Sihotang, 2015)

2.4 Halte

Menurut Keputusan DEPHUB 271/HK.105/DRJD/96 ”halte” adalah tempat perhentian kendaraan penumpang umum untuk menurunkan dan/atau menaikkan penumpang yang dilengkapi dengan bangunan. Halte secara teknis memiliki ketentuan jarak yang berbeda tergantung pada setiap tata guna lahannya, secara rinci dalam tabel 2.1 berikut :

Tabel 2.1 Ketentuan Jarak Halte (Dephub1996)

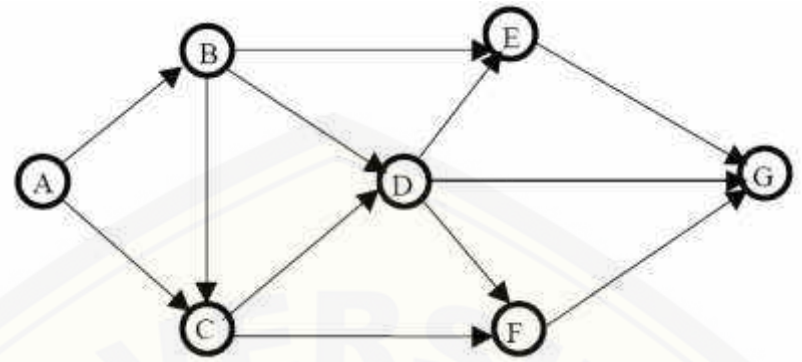
No	Tata Guna Lahan	Lokasi	Jarak Tempat Henti (m)
1	Pusat kegiatan sangat padat: pasar, pertokoan	CBD, Kota	200 – 300
2	Padat : perkantoran, sekolah, jasa	Kota	300 - 400
3	Pemukiman	Kota	300 – 400
4	Campuran padat : perumahan, sekolah, jasa	Pinggiran	300 – 500
5	Campuran jarang : perumahan, ladang, sawah, tanah kosong	Pinggiran	500 – 1000

Menurut Abubakar (1996), jenis tempat henti digolongkan menjadi 2 jenis, yaitu :

1. Tempat henti dengan lindungan (shelter), adalah tempat henti yang berupa bangunan yang digunakan penumpang untuk menunggu bus atau angkutan umum lain yang dapat melindungi dari cuaca.
2. Tempat henti tanpa lindungan (bus stop) adalah tempat henti yang digunakan untuk perhentian sementara bus atau angkutan umum lainnya pada waktu menaikkan dan menurunkan penumpang (Prima E., Rahanda J.I, 2013).

2.5 Permasalahan Rute Terpendek

Rute terpendek adalah suatu jaringan pengarahan perjalanan dimana sesorang pengarah jalan ingin menentukan rute terpendek antara dua kota, berdasarkan beberapa rute alternatif yang tersedia, dimana titik tujuan hanya satu . Gambar 2.1 menunjukkan suatu graf ABCDEFG yang berarah dan tidak berbobot (Liwang, 2013).



Gambar 2.1 Graf ABCDEFG (Liwang, 2013)

Pada gambar diatas, misalkan kita dari kota A ingin menuju Kota G. Untuk menuju kota G, dapat dipilih beberapa rute yang tersedia :

A →→ B → C → D → E → G

A → B → C → D → F → G

A → B → C → D → G

A → B → C → F → G

A → B → D → E → G

A → B → D → F → G

A → B → D → G

A → B → E → G

A → C → D → E → G

A → C → D → F → G

A → C → D → G

A → C → F → G

Berdasarkan data diatas, dapat dihitung rute terpendek dengan mencari jarak antara rute-rute tersebut. Apabila jarak antar rute belum diketahui, jarak dapat dihitung berdasarkan koordinat kota-kota tersebut, kemudian menghitung rute terpendek yang dapat dilalui (Liwang, 2013).

2.6 Algoritma Floyd Warshall

Algoritma yang ditemukan oleh Warshall untuk mencari *path* terpendek merupakan algoritma yang sederhana dan mudah implementasinya. Masukan algoritma warshall adalah matrik hubung graf berarah berbobot dan keluarannya adalah *path* terpendek dari semua titik ke semua titik. Algoritma warshall memulai iterasi dari titik awalnya kemudian memperpanjang *path* dengan mengevaluasi titik demi titik hingga mencapai titik tujuan dengan jumlah bobot yang seminimum mungkin.

Misalkan persamaan (1) W0 adalah matrik hubung graf berarah berlabel mula-mula (W) . Persamaan (3) W* adalah matrik hubung minimal dengan hasil dari persamaan (2) Wij* = *path* terpendek dari titik vi ke vj. Algoritma warshall untuk mencari *path* terpendek adalah sebagai berikut :

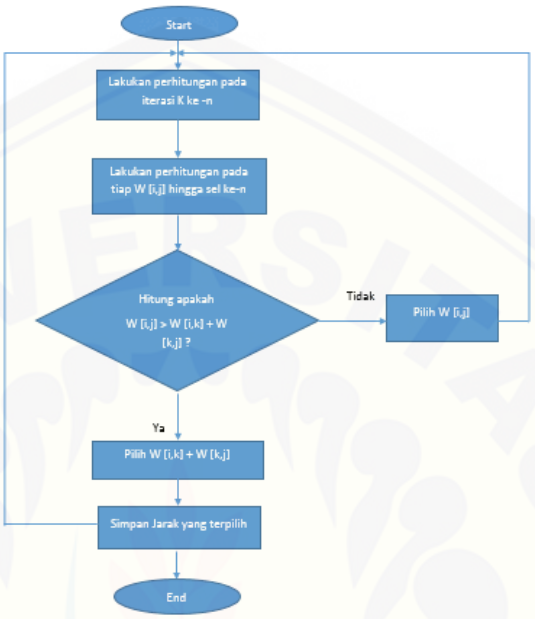
1. W = W0 .....*Persamaan. 1*
2. Untuk k = 1 hingga n, lakukan:
   Untuk i = 1 hingga n, lakukan:
   Untuk j = 1 hingga n,
   Jika W[i,j] > W[i,k] + W[k,j] maka
   Tukar W[i,j] dengan W[i,k] + W[k,j] .....*Persamaan. 2*
3. W* = W .....*Persamaan. 3*

Keterangan :

W = jarak terpendek
W0 = matrik hubung graf berarah berlabel mula-mula
k = node yang menjadi titik tengah
i = titik awal
j = titik akhir
W* = matrik hubung minimal

Dalam iterasinya untuk mencari *path* terpendek, algoritma warshall membentuk n matrik sesuai dengan iterasi k. Hal itu menyebabkan waktu prosesnya lambat,

terutama untuk n yang besar (Istyanto, 2014). Gambar 2.2 menunjukkan *flowchart* algoritma floyd warshall.



Gambar 2.2 *Flowchart* Algoritma *Floyd Warshall* (Kriswanto R.Y., Bendi R.K.J, 2014)

2.7 Aplikasi

Aplikasi merupakan penggunaan dalam suatu komputer, instruksi *(instruction)* atau pernyataan *(statement)* yang disusun sedemikian rupa sehingga komputer dapat memproses input menjadi output (Ipulhe, 2015). Definisi lain menurut Hengky W.Pramana, aplikasi adalah suatu unit perangkat lunak yang dibuat untuk melayani kebutuhan akan beberapa aktivitas seperti sistem perniagaan, game palayanan masyarakat, periklanan, atau semua proses yang hampir dilakukan manusia (Setyawan, 2012).

Penjelasan diatas dapat disimpulkan bahwa aplikasi merupakan suatu program komputer yang dibuat untuk mengerjakan dan melaksanakan tugas khusus dari pengguna.

2.8 *Global Positioning System (GPS)*

Global Positioning System (GPS) merupakan suatu kumpulan satelit dan sistem kontrol yang memungkinkan sebuah penerima GPS untuk mendapatkan lokasinya di permukaan bumi 24 jam sehari . Setiap satelit ini mengelilingi bumi sekitar 12000 mil atau 19.300 km, mengelilingi bumi 2 kali setiap harinya. Orbit satelit – satelit ini diatur sedemikian rupa sehingga pada setiap saat, suatu tempat di bumi akan dijangkau minimal 4 satelit. GPS terdiri dari 3 bagian penting, yaitu :

1. Kontrol, yaitu tanggung jawab untuk mengawasi kinerja satelit pada orbitnya.
2. Ruang, yaitu ruang untuk satelit beroperasi. Bisa disebut dengan orbit. Terdapat 24 satelit yang mengelilingi orbit.
3. Pengguna, yaitu bagian dari sistem karena pengguna adalah orang yang menggunakan layanan sistem GPS sehingga sistem ini bekerja. Gambar 2.3 merupakan ilustrasi dari sistem GPS.

Gambar 2.3 Satelit GPS

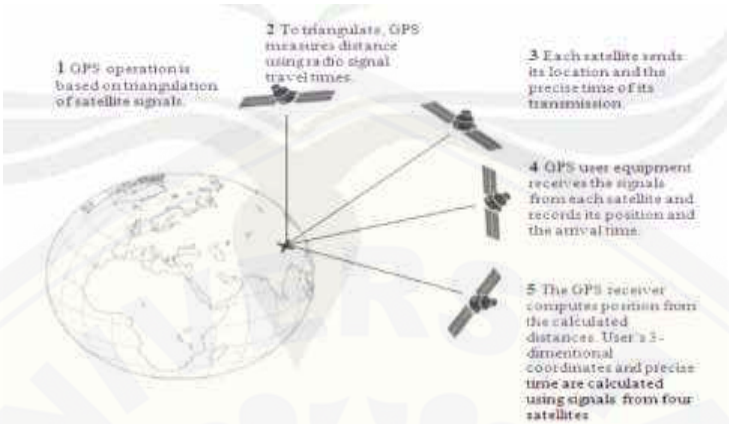
Dua Puluh Empat (24) satelit yang membentuk jaringan itu mengorbit setinggi 12.000 mil. Secara konstan bergerak, mengorbit mengelilingi Bumi dua kali per harinya (Orbit Periode setiap 12 Jam) dengan kecepatan ±7.000 km per jam. Orbit dari satelit – satelit ini didesain sedemikian rupa sehingga kapanpun, dimanapun di permukaan bumi, *Receiver* GPS akan dapat mengakses paling tidak 4 satelit. Daya listrik untuk satelit GPS dimemakai energi matahari dengan solar panel, selain itu baterai cadangan tersedia agar satelit bisa tetap bekerja saat gerhana matahari atau saat

energi matahari tidak tersedia. Navstar GPS *system* dikontrol oleh 5 *Master ground* Stasiun yang secara khusus memonitor satelit – satelit GPS di angkasa yang berlokasi di Hawaii, Ascension Island, Diego Garcia, Kwajalein, dan Colorado Spring.

Cara kerja GPS seperti pada gambar 2.4, satelit GPS mengelilingi bumi dua kali sehari dalam orbit yang amat presisi sambil memancarkan sinyal ke bumi. GPS *receiver* menerima informasi ini menggunakan metode Triangulasi untuk menghitung secara pasti di mana lokasi *receiver*. Pada dasarnya, *receiver* membandingkan *timing* dalam *micro second* pulsa waktu dari sinyal yang ditransmisikan oleh satelit dengan *timing* pulsa waktu, yang diterima pada *receiver* dengan transmisi *pseudo random code*. Perbedaan waktu inilah yang akan memberitahu *receiver* seberapa jauh dan arah satelit berada darinya. Setelah jarak diukur dengan sejumlah satelit GPS lainnya, *receiver* bisa menentukan posisinya dalam koordinat lintang dan bujur derajat. *Receiver* harus mengunci paling tidak 3 satelit untuk menghitung posisi 2 dimensi (garis lintang dan garis bujur) dan lintasan pergerakan. Dengan 4 atau lebih satelit yang dapat diakses, *receiver* dapat menentukan posisi 3 dimensi (+ ketinggian). Sekali posisi dari pengguna dapat ditentukan, *receiver* GPS dapat juga menentukan informasi lain seperti kecepatan, lintasan yang telah dilewati, jarak perjalanan yang sudah ditempuh, jarak ke tempat tujuan, waktu *sunrise* dan *sunset* dan lain sebagainya (Gunawan, 2013).

Metode triangulasi inilah yang digunakan satelit GPS untuk menentukan titik lokasi *receiver.* Untuk mendukung perhitungan triangulasi, *receiver* harus mengetahui dua hal :

1. Lokasi dari paling tidak 3 satelit yg dapat di akses
2. Jarak antara Anda dengan satelit-satelit tersebut.



Gambar 2.4 Menggambarkan cara kerja GPS (*Mosque Tracking on Mobile GPS and Prayer Times Synchronization for Unfamiliar Area*, 2011)

2.9 Google *Maps*

Google Maps adalah layanan aplikasi peta *online* yang disediakan oleh *Google* secara gratis. Layanan peta *Google Maps* secara resmi dapat diakses melalui situs http://maps.google.com. Pada situs tersebut dapat dilihat informasi geografis pada hampir semua permukaan di bumi kecuali daerah kutub utara dan selatan. Layanan ini dibuat sangat interaktif, karena di dalamnya peta dapat digeser sesuai keinginan pengguna, mengubah level *zoom*, serta mengubah tampilan jenis peta. *Google Maps* mempunyai banyak fasilitas yang dapat dipergunakan misalnya pencarian lokasi dengan memasukkan kata kunci, kata kunci yang dimaksud seperti nama tempat, kota, atau jalan, fasilitas lainnya yaitu perhitungan rute perjalanan dari satu tempat ke tempat lainnya (Siswanto, 2012).

2.9.1 Google *Maps* API

API atau *Application Programming Interface* merupakan suatu dokumentasi yang terdiri dari *interface*, fungsi, kelas, struktur dan sebagainya untuk membangun sebuah perangkat lunak. Dengan adanya *API* ini, maka memudahkan programmer untuk "membongkar" suatu *software* untuk kemudian dapat dikembangkan atau

diintegrasikan dengan perangkat lunak yang lain. *API* dapat dikatakan sebagai penghubung suatu aplikasi dengan aplikasi lainnya yang memungkinkan programmer menggunakan sistem *function*. Proses ini dikelola melalui *operating system*. Keunggulan dari *API* ini adalah memungkinkan suatu aplikasi dengan aplikasi lainnya dapat saling berhubungan dan berinteraksi. Bahasa pemrograman yang digunakan oleh *Google Maps* yang terdiri dari *HTML, Javascript* dan *AJAX* serta *XML*, memungkinkan untuk menampilkan peta *Google Maps* di *website* lain.

Google juga menyediakan layanan *Google Maps API* yang memungkinkan para pengembang untuk mengintegrasikan *Google Maps* ke dalam *web*site masingmasing dengan menambahkan data *point* sendiri. Dengan menggunakan *Google Maps API*, *Google Maps* dapat ditampilkan pada *web* site eksternal. Agar aplikasi *Google Maps* dapat muncul di *website* tertentu, diperlukan adanya *API key*. *API key* merupakan kode unik yang digenerasikan oleh google untuk suatu *website* tertentu, agar *server Google Maps* dapat mengenali (Siswanto, 2012).

2.10 Android

Android adalah sistem operasi berbasis Linux yang dirancang untuk perangkat seluler layar sentuh seperti telepon pintar dan komputer tablet. Android awalnya dikembangkan oleh Android, Inc., dengan dukungan finansial dari Google, yang kemudian membelinya pada tahun 2005. Sistem operasi ini dirilis secara resmi pada tahun 2007, bersamaan dengan didirikannya Open Handset Alliance, konsorsium dari perusahaan-perusahaan perangkat keras, perangkat lunak, dan telekomunikasi yang bertujuan untuk memajukan standar terbuka perangkat seluler. Ponsel Android pertama mulai dijual pada bulan Oktober 2008.

Android merupakan sistem operasi dengan sumber terbuka, dan Google merilis kodenya di bawah Lisensi Apache. Kode dengan sumber terbuka dan lisensi perizinan pada Android memungkinkan perangkat lunak untuk dimodifikasi secara bebas dan didistribusikan oleh para pembuat perangkat, operator nirkabel, dan pengembang

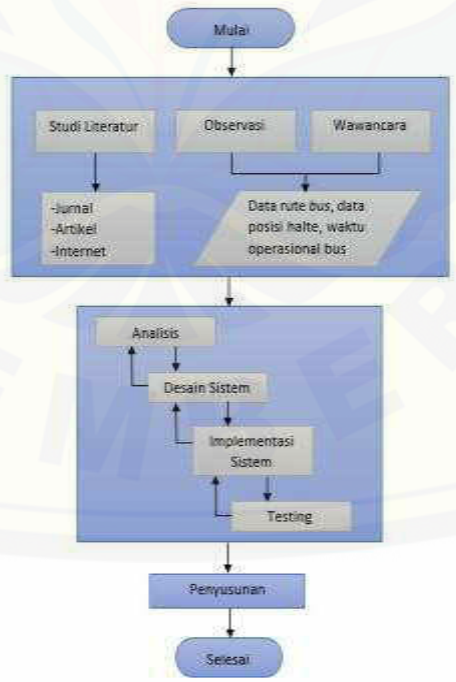
aplikasi. Selain itu, Android memiliki sejumlah besar komunitas pengembang aplikasi (apps) yang memperluas fungsionalitas perangkat, umumnya ditulis dalam versi kustomisasi bahasa pemrograman Java. Pada bulan Oktober 2012, ada sekitar 700.000 aplikasi yang tersedia untuk Android, dan sekitar 25 juta aplikasi telah diunduh dari Google Play, toko aplikasi utama Android. Sebuah survey pada bulan April-Mei 2013 menemukan bahwa Android adalah platform paling populer bagi para Android juga menjadi pilihan bagi perusahaan teknologi yang menginginkan sistem operasi berbiaya rendah, bisa dikustomisasi, dan ringan untuk perangkat berteknologi tinggi tanpa harus mengembangkannya dari awal. Sifat Android yang terbuka telah mendorong munculnya sejumlah besar komunitas pengembang aplikasi untuk menggunakan kode sumber terbuka sebagai dasar proyek pembuatan aplikasi, dengan menambahkan fitur-fitur baru bagi pengguna tingkat lanjut atau mengoperasikan Android pada perangkat yang secara resmi dirilis dengan menggunakan sistem operasi lain (Safaat N, 2012)

BAB 3. METODOLOGI PENELITIAN

Metodologi penelitian adalah sekumpulan peraturan, kegiatan, dan prosedur yang digunakan oleh pelaku suatu disiplin ilmu. Metodologi juga merupakan analisis teoritis mengenai suatu cara atau metode. Metode penelitian merupakan langkah dan prosedur yang akan dilakukan dalam mengumpulkan data atau informasi empiris guna memecahkan permasalahan, dan mengumpulkan informasi yang diperlukan untuk menyusun penelitian ini.

3.1 Alur Penelitian

Alur penelitian merupakan urutan langkah penelitian yang dilakukan mulai dari studi literatur, pengumpulan data, pengolahan data, perancangan sistem, dan pengimplementasian rancangan sistem. Diagram alur pada penelitian ini dapat dilihat pada Gambar 3.1 berikut.



Gambar 3.1 Diagram Alur Penelitian

Pada penelitian ini digunakan dua jenis penelitian, yaitu penelitian kualitatif dan penelitian kuntitatif. Jenis penelitian kualitatif digunakan karena penelitian ini menganalisa studi literatur dan melakukan *interview* untuk pengumpulan sampel data dan jenis penelitian kuantitatif digunakan karena dalam penelitian ini menerapkan serta mengkaji teori yang sudah ada sebelumnya.

3.2 Jenis Penelitian

Penelitian ini menggunakan jenis penelitian kualitatif dan jenis penelitian kuantitatif. Menggunakan penelitian jenis kualitatif pada proses menganalisa studi literatur yang berhubungan dengan aplikasi pencarian halte terdekat *bus* Trans SARBAGITA. Menggunakan penelitian jenis kuantitatif karena data yang diolah dalam bentuk angka.

3.3 Pengumpulan Data

Teknik pengumpulan data merupakan langkah penting dalam melakukan sebuah riset, karena tujuan utama dari tahapan ini adalah mendapatkan data. Dalam tahap pengumpulan data, penulis menjabarkan beberapa langkah sebagai berikut :

a. Wawancara (*Interview)*

Wawancara yang dilakukan yaitu melalui komunikasi dua arah antara peneliti dengan informan untuk mendapatkan data yang *valid* dalam penelitian. Dalam penelitian ini penulis melakukan wawancara secara langsung kepada seorang narasumber. Dari hasil wawancara tersebut, dapat diperoleh data *valid* yang dapat digunakan dalam penelitian. Data yang dibutuhkan adalah berupa data operasional *bus* seperti jumlah koridor, nama halte, denah koridor, dan rute *bus*

b. Pengamatan *(*Observasi*)*

Observasi yaitu melakukan pengamatan secara langsung ke objek-objek yang sedang diteliti. Observasi ini bertujuan untuk dapat mengetahui langsung bagaimana alur kerja yang terjadi pada objek yang diteliti. Hasil dari observasi langsung ke objek ini yang dapat digunakan dalam penelitian berupa titik koordinat letak halte yang ada.

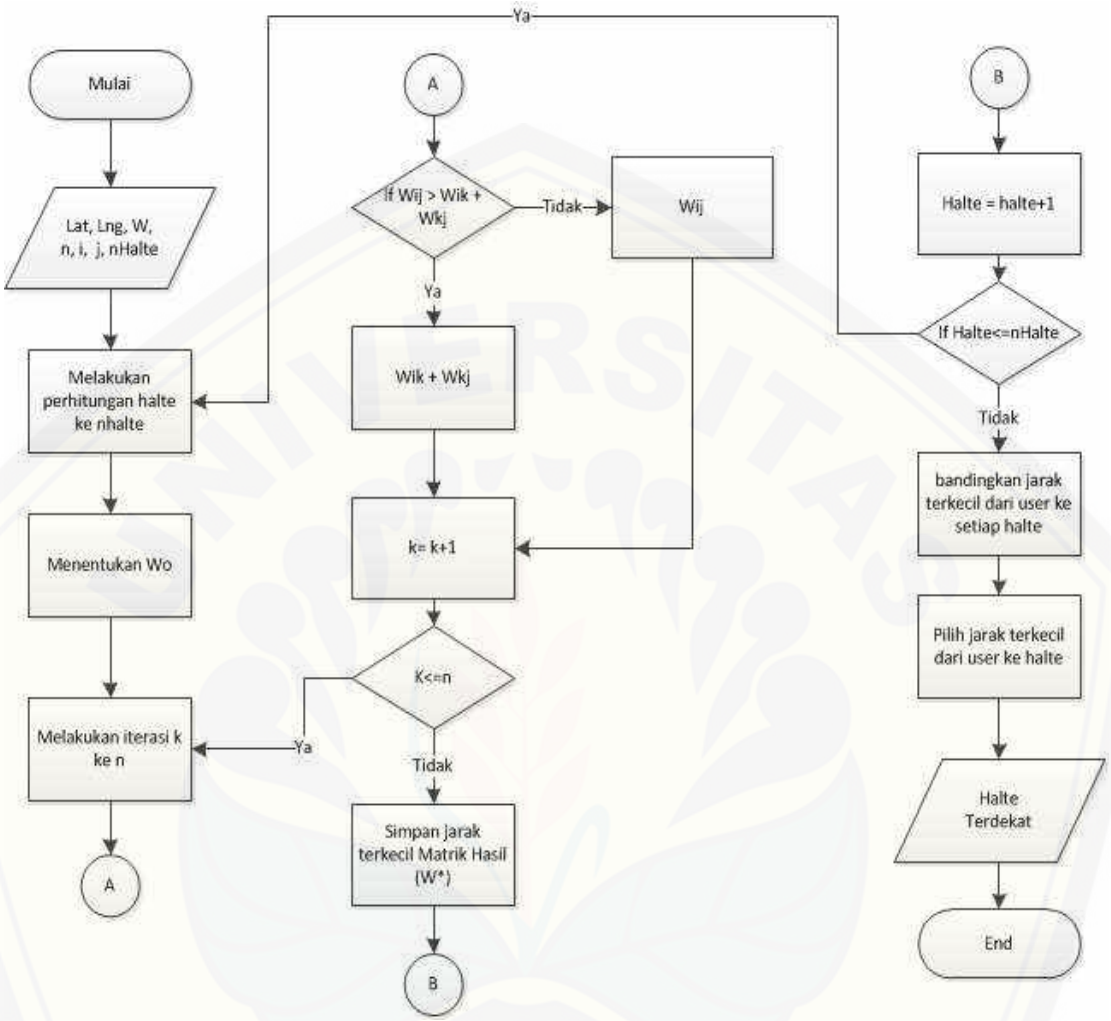
c. Studi Pustaka

Pengumpulan data terkait penelitian ini adalah didapat dari buku literatur, jurnal online, *ebook*, serial online, dan observasi yang berkaitan dengan perancangan dan pembuatan sistem dan pembahasan ini.

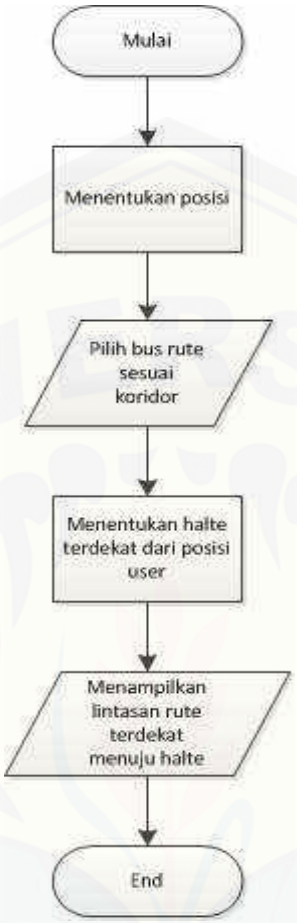
3.4 Analisis Data

Tahap analisis data dimulai dengan menelaah data yang telah dikumpulkan. Data yang telah terkumpul akan dimasukan sebagai parameter dalam perhitungan algoritma *Floyd Warshall*. Parameter yang digunakan merupakan jarak yang didapat dari analisis data. Langkah selanjutnya memasukkan data yang telah didapat ke dalam perhitungan algoritma *Floyd Warshall.*

Menu dalam aplikasi pencarian halte terdekat *bus* Trans SARBAGITA ini terdapat 2 yaitu menu *my location* dan *bus route* (koridor). Menu *my location* digunakan pengguna untuk mengetahui letak posisinya saat ini dan menu *bus route* digunakan pengguna untuk mencari letak halte terdekat dari posisinya saat ini sesuai koridor yang dipilih. Berdasarkan dari sistem yang nantinya akan dibuat maka *flowchart* penerapan *Floyd Warshall* untuk menemukan halte terdekat dapat dilihat pada Gambar 3.2 dan flowchart aplikasi pencarian halte terdekat *bus* Trans SARBAGITA dapat dilihat pada Gambar 3.3.



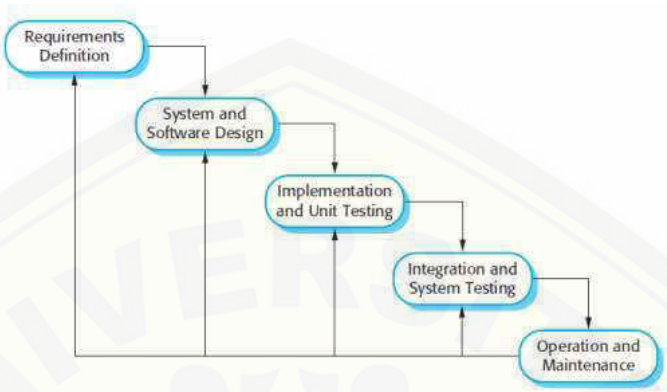
Gambar 3.2 *Flowchart* Penerapan Algoritma *Floyd Warshall* dalam Aplikasi Trans SARBAGITA *Bus Stop Finder*



Gambar 3.3 *Flowchart* Aplikasi Trans SARBAGITA *Bus Stop Finder*

3.5 Pembuatan Sistem

Metode yang digunakan dalam pembuatan dan pengembangan dalam sistem ini yaitu SDLC (*System Development Life Cycle*) dengan model *Waterfall* karena sistem ini tergolong sistem bersekala kecil. Model *waterfall* merupakan metode yang sistematik dan sekuensial yang mulai pada tingkat dan kemajuan sistem sampai pada analisis, desain, kode, *test* dan pemeliharaan (Roger S. Pressman : 2002). Metode pengembangan sistem ini memiliki beberapa kegiatan yang dapat dikelompokkan menjadi beberapa tahap, menurut Ian Sommerville (2003:23) dapat dilihat pada Gambar 3.4.



Gambar 3.4 *Waterfall* Model

3.5.1 Analisis Kebutuhan

Tahap analisa kebutuhan adalah tahap mencari data yang dibutuhkan dalam pembuatan sistem dalam proses ini dilakukan dengan cara melakukan penelitian. Dalam penelitian kualitatif, analisis data dilakukan selama dan setelah pengumpulan data. Pada bagian analisis data diuraikan pengaturan secara sistematis transkrip-transkrip wawancara, catatan lapangan dan bahan-bahan lain agar peneliti dapat menyajikan temuannya. Analisis ini melibatkan pengerjaan, pengorganisasian, pemecahan dan sintesis data serta pencarian pola, pengungkapan hal yang penting, dan penentuan apa yang dilaporkan.

Data yang didapat dari pengamatan dan observasi diolah untuk mendapatkan solusi agar dapat menghitung dengan algoritma *Floyd Warshall.* Data koordinat posisi *user* didapat dari metode *location based service* yang bekerja menggunakan *GPS* pada *mobile device.* Perhitungan menggunakan parameter jarak yang didapat dari titik koordinat posisi *user* dan titik koordinat halte. Data kuantitatif tersebut diterapkan pada algoritma *Floyd Warshall* untuk menemukan halte terdekat dengan koridor yang dipilih.

3.5.2 Desain Sistem

Tahapan kedua dari pemodelan *waterfall* adalah desain sistem. Proses pembuatan desain sistem pada penelitian ini menggunakan *Unified Modeling Languange* (UML) yang dirancang menggunakan konsep *Object-Oriented Programming* (OOP). Pemodelan *Unified Modeling Languange* (UML) yang digunakan adalah sebagai berikut :

1. *Use Case Diagram*

Use Case Diagram merupakan model atau *diagram* yang digunakan untuk menggambarkan kebutuhan fungsional yang diharapkan dari suatu sistem. Umumnya *use case diagram* menekankan pada "siapa" melakukan "apa" dalam *enviroment* pada suatu sistem yang dibangun. *Use case diagram* digambarkan dari beberapa *actor, use case,* dan interaksi diantara komponen – komponen tersebut yang dapat memberikan informasi dari suatu sistem yang akan dibangun.

2. *Use Case Scenario*

Use Case Scenario merupakan deskripsi atau penjabaran alur kinerja (*step – step* dari tiap *use case*) dari *use case diagram* yang telah dibuat. Umumnya *use case scenario* digambarkan dalam bentuk tabel yang dapat menggambarkan penjabaran alur kinerja dari tiap *use-case* yang ada.

3. *Activity Diagram*

Activity Diagram merupakan model atau *diagram* yang menggambarkan aktivitas *(activity)* dari suatu sistem yang akan dibangun. Sehingga, dengan *activity diagram*, *process* dari sistem yang akan dibangun dapat diketahui dengan jelas berdasarkan aktivitasnya saat adanya suatu aksi atau *action* pada sistem.

4. *Sequence Diagram*

Sequence Diagram merupakan model atau *diagram* yang menggambarkan interaksi antar objek yang mengindikasikan komunikasi diantara obyek - obyek tersebut di dalam sistem yang disusun pada sebuah urutan atau rangkaian waktu. Interaksi antar objek tersebut terdiri dari pengguna *(user)*, tampilan *(display)*, dan lain

sebagainya berupa pesan *(message)*. Sehingga, dengan *sequence diagram*, aliran logika dalam sebuah sistem dapat dimodelkan secara *visual* dalam bentuk *diagram.*

5. *Class Diagram*

Class Diagram merupakan model statis yang menggambarkan struktur dan deskripsi *class* serta hubungannya antara *class*. *Class diagram* mirip dengan ERD pada perancangan *database*, bedanya pada ERD tidak terdapat operasi atau *method* tetapi hanya atribut saja. *Class Diagram* terdiri dari nama kelas, atribut dan operasi atau *method.*

6. Entity Relation Diagram (ERD)

Entity relation diagram (ERD) adalah *diagram* yang menggambarkan relasi objek-objek dasar data dalam sebuah basis data.

3.5.3 Penulisan Kode Program

Implementasi sistem merupakan tahap untuk mengimplementasikan atau mengubah desain sistem yang telah dibuat kedalam kode program. Tahap pertama yang dilakukan dalam implementasi adalah penulisan kode program (*coding*) menggunakan bahasa pemrograman java dan layout menggunakan XML (*eXtended Markup Languange*). Pemograman dilakukan menggunakan Android Studio. Kemudian untuk tahap kedua mendapatkan basis data dari *server* Google *Maps* API menggunakan PHP dan data dipaketkan menggunakan JSON karena digunakan untuk aplikasi android.

3.5.4 Pengujian Sistem

Tahap testing harus dilakukan sebelum sistem diserahkan kepada *user*. Tahap dilakukan agar *programe*r dapat mengetahui apakah sistem yang dibangun sesuai dengan kebutuhan yang telah dianalisis diawal. Serta agar mengetahui apakah terdapat kesalahan pada sistem yang dibangun. Tahap *testing* dilakukan guna menyempurnakan sistem sebelum diserahkan kepada *user*. Pada tahap *testing* ini dilakukan pengujian dengan metode *white box* dan metode *black box*.

1. White Box Testing

White box testing merupakan pengujian pada modul pengkodean program yang dilakukan oleh peneliti. Pengujian ini dilakukan dengan menghitung *independent path* dengan menggunakan *cyclomatic complexity*. Untuk menghitung *cyclomatic complexity* digunakan rumus pada Persamaan (4) :

$$V(G) = E - N + 2 \text{.....} \textit{Persamaan. 4}$$

Keterangan :

E = jumlah *edge* grafik alir

N = jumlah *node* grafik alir

2. Black Box Testing

Black box testing merupakan pengujian yang menitik beratkan pada uji fungsionalitas dari program yang dibuat. Pengujian ini digunakan untuk menemukan ketidaksesuaian program dengan kebutuhan fungsional maupun non-fungsional. Hal yang perlu dilakukan dalam pengujian ini adalah menguji *interface* dari program untuk memastikan suatu masukan diproses oleh sistem dengan benar dan menghasilkan keluaran yang sesuai dengan perancangan.

3.5.5 Penerapan Program

Aplikasi yang telah melewati masa pengujian dapat diimplementasikan pada *smartphone* yang berbasis android dan dapat digunakan dengan keadaan tersambung ke jaringan internet serta menyalakan GPS pada perangkat *smarthphone*.

BAB 4. DESAIN DAN PRANCANGAN

Bab ini akan menjelaskan mengenai proses perancangan untuk mengimplementasikan algoritma *Floyd Warshall* untuk mencari halte terdekat *bus* Trans SARBAGITA berbasis android. Proses perancangan sistem dimulai dari analisis kebutuhan fungsional dan non-fungsional sistem, dilanjutkan dengan pembuatan *use case diagram,* skenario, *activity diagram, sequence diagram, class diagram* dan *entity relation diagram* (ERD).

4.1 Analisis Kebutuhan Perangkat Lunak

Aplikasi yang dirancang merupakan aplikasi pencarian halte terdekat *bus* Trans SARBAGITA di provinsi Bali. Aplikasi ini bertujuan untuk mempermudah para pengguna menemukan lokasi halte terdekat dari posisinya. Dalam perancangan aplikasi ini terdapat batasan-batasan. Aplikasi yang penulis buat ini berhubungan dengan pemetaan lokasi halte *bus* dan rute untuk menuju lokasi halte terdekat.

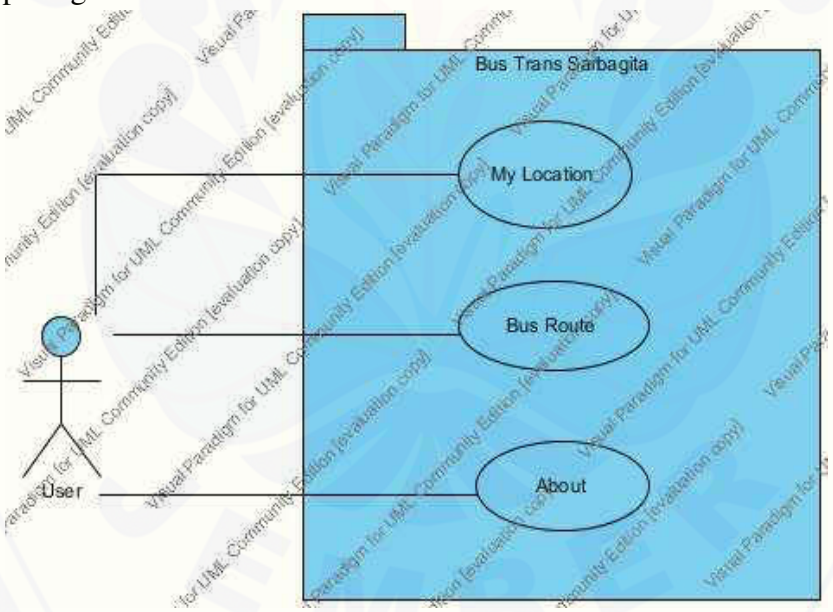
Aplikasi Trans SARBAGITA *Bus Stop Finder* merupakan aplikasi untuk mencari halte terdekat *bus* berbasis android menggunakan metode algoritma *Floyd Warshall.* Aplikasi ini dapat digunakan pengguna *bus* Trans SARBAGITA untuk mencari tahu letak halte terdekat dari posisi *user* saat itu. Aplikasi ini dapat memberikan rute terdekat menuju halte terdekat. Adapun kebutuhan fungsional dan non-fungsional dari sistem ini adalah sebagai berikut :

1. Kebutuhan Fungsional Sistem
   a. Sistem mampu menampilkan posisi *user.*
   b. Sistem mampu menampilkan halte terdekat dari posisi *user* sesuai koridor yang dipilih.
   c. Sistem mampu menampilkan pemetaan halte *bus* berdasarkan koridor.
   d. Sistem mampu menampilkan rute terdekat dari posisi *user* ke halte terdekat.

2. Kebutuhan Non Fungsional Sistem
   a. Sistem dirancang dengan tampilan *user friendly* untuk memudahkan dalam penggunaan aplikasi.
   b. Aplikasi dapat digunakan pada *smartphone* android versi 4.0 ke atas.
   c. Aplikasi dapat diakses 24 jam.

4.2 *Use case Diagram*

Use case diagram adalah gambaran fungsionalitas dari suatu sistem, sehingga *user* atau pengguna sistem paham dan mengerti mengenai kegunaan sistem yang akan dibangun. Diagram ini akan menjelaskan fitur-fitur dan aktor siapa saja dalam aplikasi pencarian halte terdekat *bus* Trans SARBAGITA. *Use case diagram* aplikasi dapat dilihat pada gambar 4.1.



Gambar 4.1 *Use case* Aplikasi Trans SARBAGITA *Bus Stop Finder*

Definisi *use case* pada *use case* aplikasi pencarian halte terdekat dapat dilihat pada Tabel 4.1 dan untuk definisi dari aktor yang terdapat pada *use case* aplikasi pencarian halte terdekat bisa dilihat pada tabel 4.2.

Tabel 4.1 Definisi *Use Case* Aplikasi Pencarian Halte Terdekat

No	*Usecase*	Deskripsi
1	*My Location*	Proses untuk mengetahui letak atau posisi *user* saat menggunakan aplikasi
2	*Bus Route*	Proses untuk mencari halte terdekat dengan memilih koridor terlebih dahulu
3	*About*	Menampilkan keterangan mengenai aplikasi

Tabel 4.2 Definisi Aktor Aplikasi Pencarian Halte Terdekat

No	Actor	Deskripsi
1	*User*	Pengguna *Bus* Trans SARBAGITA yang mengoperasikan aplikasi Pencarian Halte Terdekat untuk mencari halte terdekat dan informasi rute dari *Bus* Trans SARBAGITA

4.3 Skenario

Skenario berfungsi untuk menggambarkan alur sistem beserta alternatif yang akan dijalankan oleh *user* pada aplikasi pencarian halte terdekat *bus* Trans SARBAGITA. Skenario sistem dapat dilihat pada Tabel 4.3 sampai Tabel 4.5.

Tabel 4.3 Skenario *My Location*

Usecase Name	*My Location*
Usecase ID	UCD-01
Primary Actor	*User*
Brief Description	*User* mencari posisi saat ini menggunakan aplikasi

Precondition	*User* akan mencari tahu posisi saat ini
Postcondition	*User* telah mengetahui posisi saat ini
Flow of Events	
Skenario Normal UCD-01	
Aksi Aktor	**Reaksi Sistem**
1. Membuka aplikasi	
	2. Menampilkan *splash screen*
	3. Menampilkan map
4. Tekan navigasi menu	
	5. Menampilkan menu
6. Tekan *My Location*	
	7. Menampilkan posisi *user* pada map
Skenario Alternatif : Tidak Ada Koneksi Internet	
	3. Menampilkan map
	4. Menampilkan *message box "Failed to find alternate paths, check your internet connection"*
Skenario Normal : GPS Tidak Aktif (diaktifkan)	
	3. Menampilkan map

	4. Menampilkan *alert dialog* "*You must enable your GPS, enable now ?"*
5. Tekan *Yes*	
	6. Menampilkan posisi *user* pada map
Skenario Alternatif : GPS Tidak Aktif (non-aktifkan)	
	4. Menampilkan *alert dialog* "*You must enable your GPS, enable now ?"*
5. Tekan No	
	6. Menampilkan map *default* (jember)

Tabel 4.4 Skenario *Bus Route*

Usecase Name	*Bus Route*
Usecase ID	UCD-02
Primary Actor	*User*
Brief Description	*User* halte terdekat sesuai koridor
Precondition	*User* akan mencari tahu halte terdekat dari posisinya sesuai koridor pilihan
Postcondition	*User* telah mencari tahu halte terdekat dari posisinya sesuai koridor pilihan

Flow of Events	
Skenario Normal UCD-02	
Aksi Aktor	**Reaksi Sistem**
1. Membuka aplikasi	
	2. Menampilkan *splash screen*
	3. Menampilkan map berisi posisi *user* saat ini
4. Tekan navigasi menu	
	5. Menampilkan halaman menu
6. Tekan *Bus Route* (salah satu rute yang ada pada koridor 1 atau 2)	
	7. Menampilkan halte terdekat beserta lintasan rute terdekat
Skenario Alternatif : Tidak Ada Koneksi Internet	
	3. Menampilkan map
	4. Menampilkan *message box "Failed to find alternate paths, check your internet connection"*
Skenario Normal : GPS Tidak Aktif (diaktifkan)	
	3. Menampilkan map

	4. Menampilkan *alert dialog* "*You must enable your GPS, enable now ?"*
5. Tekan *Yes*	
	6. Menampilkan posisi *user* pada map
Skenario Alternatif : GPS Tidak Aktif (non-aktifkan)	
	4. Menampilkan *alert dialog* "*You must enable your GPS, enable now ?"*
5. Tekan No	
	6. Menampilkan map *default* (jember)
Skenario Alternatif : Loading Data Halte	
6. Tekan *Bus Route* (salah satu rute yang ada pada koridor 1 atau 2)	
	7. Menampilkan *message box "Please wait until all halte loaded"*

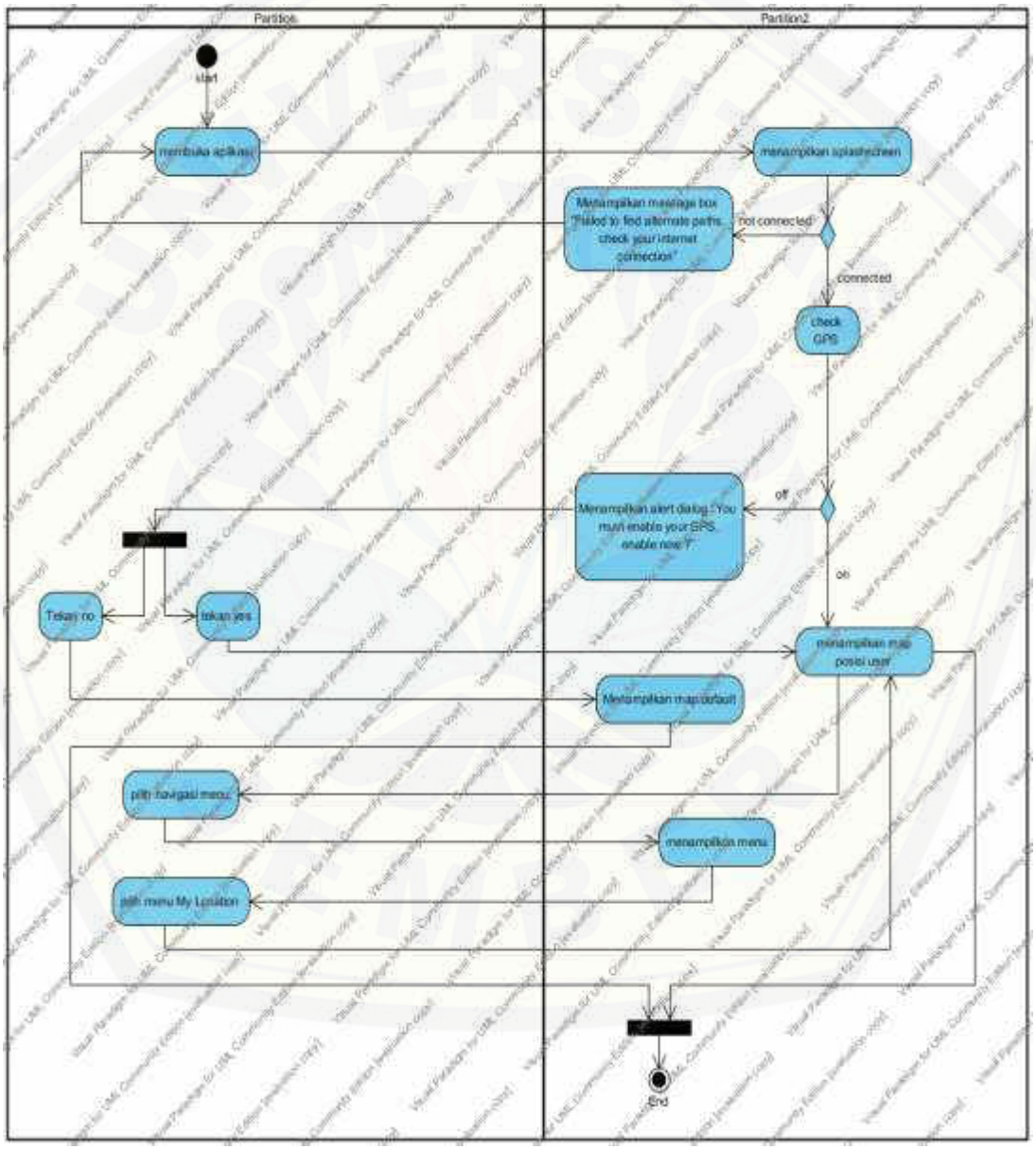
Tabel 4.5 Skenario *About*

Usecase Name	*About*
Usecase ID	UCD-03
Primary Actor	*User*
Brief Description	*User* melihat keterangan tentang aplikasi
Precondition	*User* akan melihat keterangan tentang aplikasi
Postcondition	*User* telah melihat keterangan tentang aplikasi
Flow of Events	
Skenario Normal UCD-04	
Aksi Aktor	**Reaksi Sistem**
1. Membuka aplikasi	
	2. Menampilkan *splash screen*
	3. Menampilkan map berisi posisi *user* saat ini
4. Tekan navigasi menu	
	5. Menampilkan halaman menu
6. Tekan menu *about*	

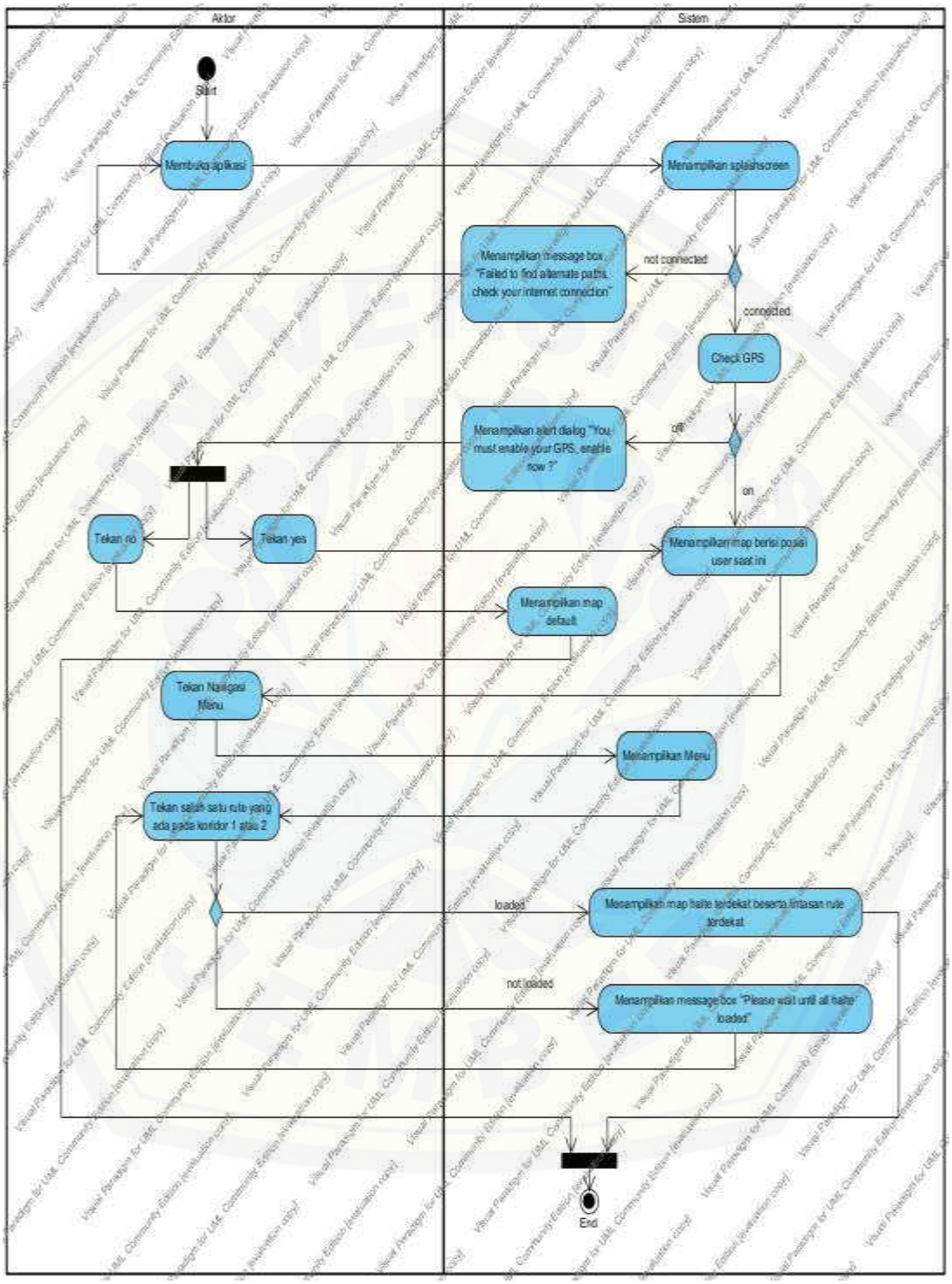
	7. Menampilkan halaman *about*
8. Tekan tombol back (X)	
	9. Menampilkan map berisi posisi *user* saat ini
Skenario Alternatif : Tidak Ada Koneksi Internet	
	3. Menampilkan map
	4. Menampilkan *message box "Failed to find alternate paths, check your internet connection"*
Skenario Normal : GPS Tidak Aktif (diaktifkan)	
	3. Menampilkan map
	4. Menampilkan *alert dialog* "*You must enable your GPS, enable now ?"*
5. Tekan *Yes*	
	6. Menampilkan posisi *user* pada map
Skenario Alternatif : GPS Tidak Aktif (non-aktifkan)	
	4. Menampilkan *alert dialog* "*You must enable your GPS, enable now ?"*
5. Tekan No	
	6. Menampilkan map *default* (jember)

4.4 *Activity Diagram*

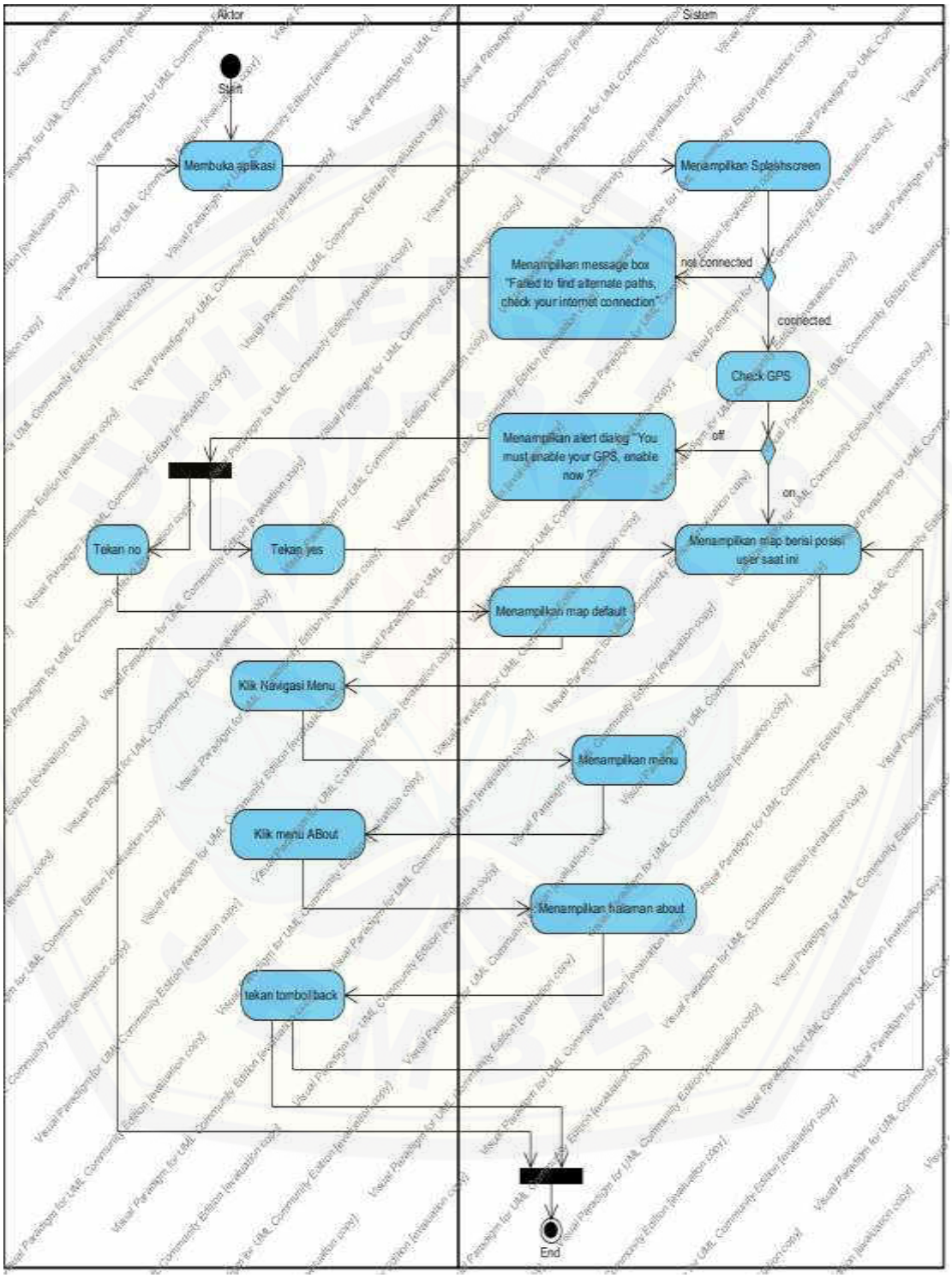
Activity diagram berfungsi untuk menggambarkan alur aktivitas yang akan dijalankan aplikasi pencarian halte terdekat *bus* Trans SARBAGITA dengan *user* dalam bentuk diagram aktivitas. *Activity Diagram* pada aplikasi dapat dilihat pada Gambar 4.2 sampai Gambar 4.4.



Gambar 4.2 *Activity Diagram My Location*



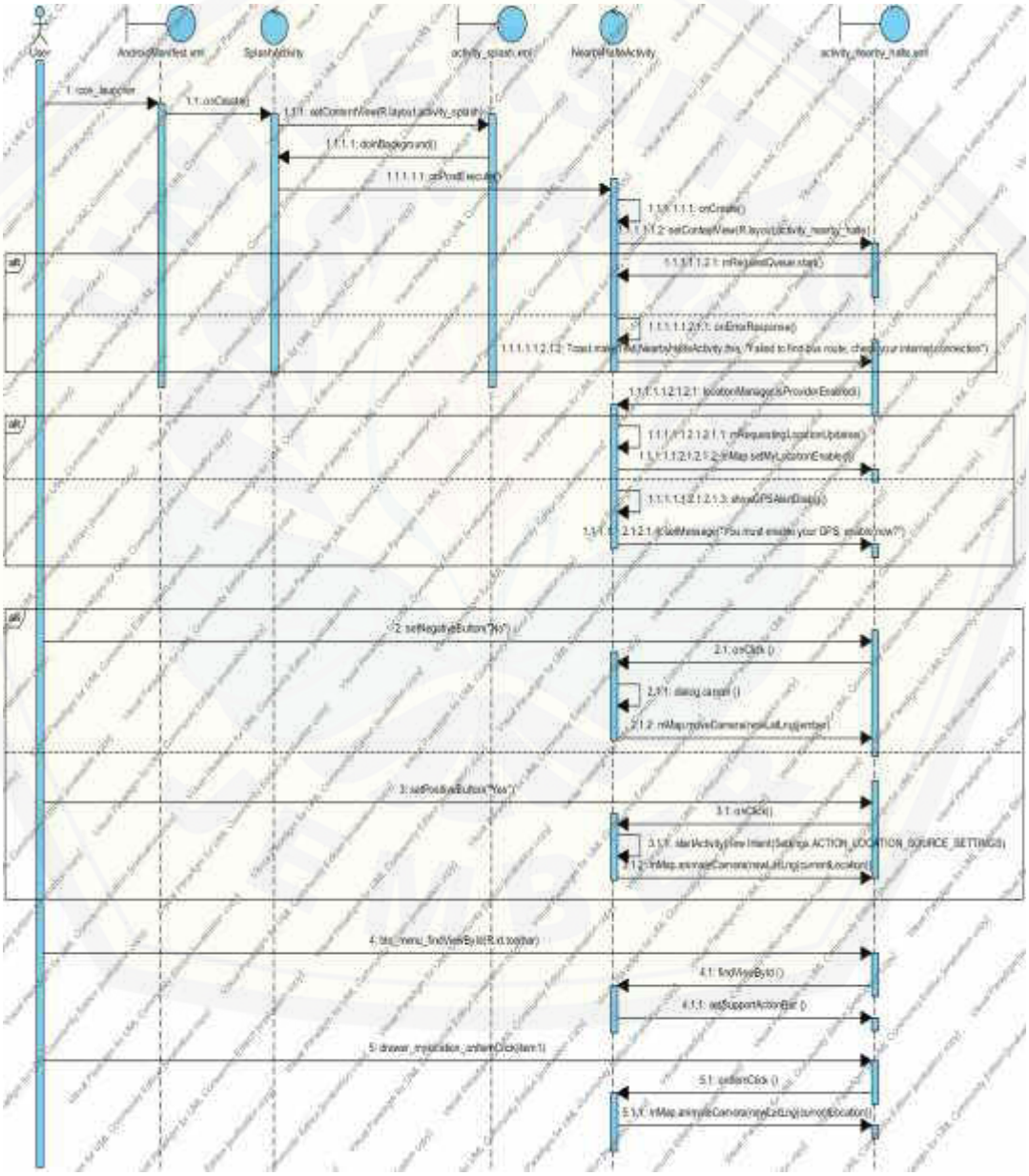
Gambar 4.3 *Activity Diagram Bus Route*



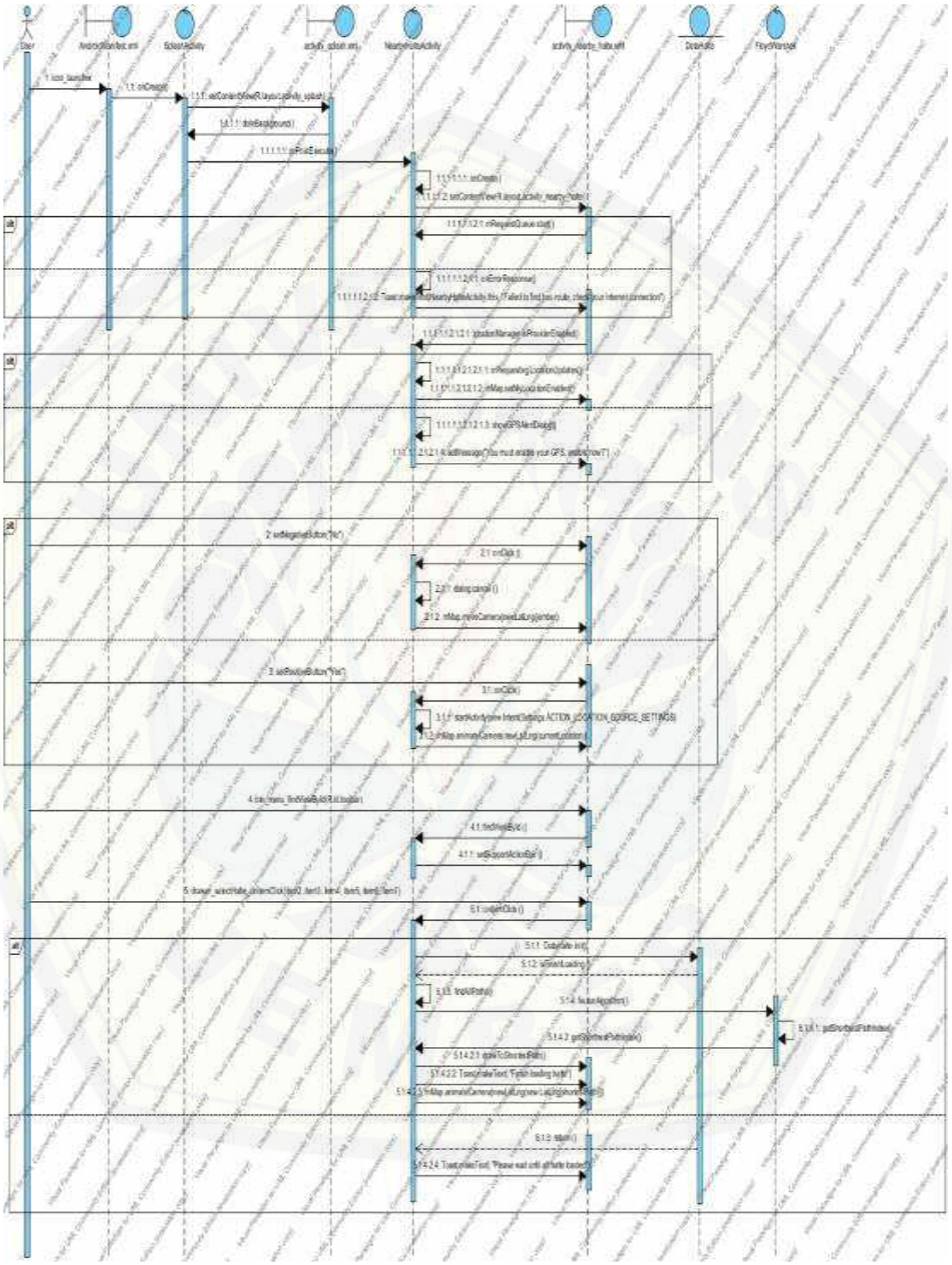
Gambar 4.4 *Activity Diagram About*

4.5 *Sequence Diagram*

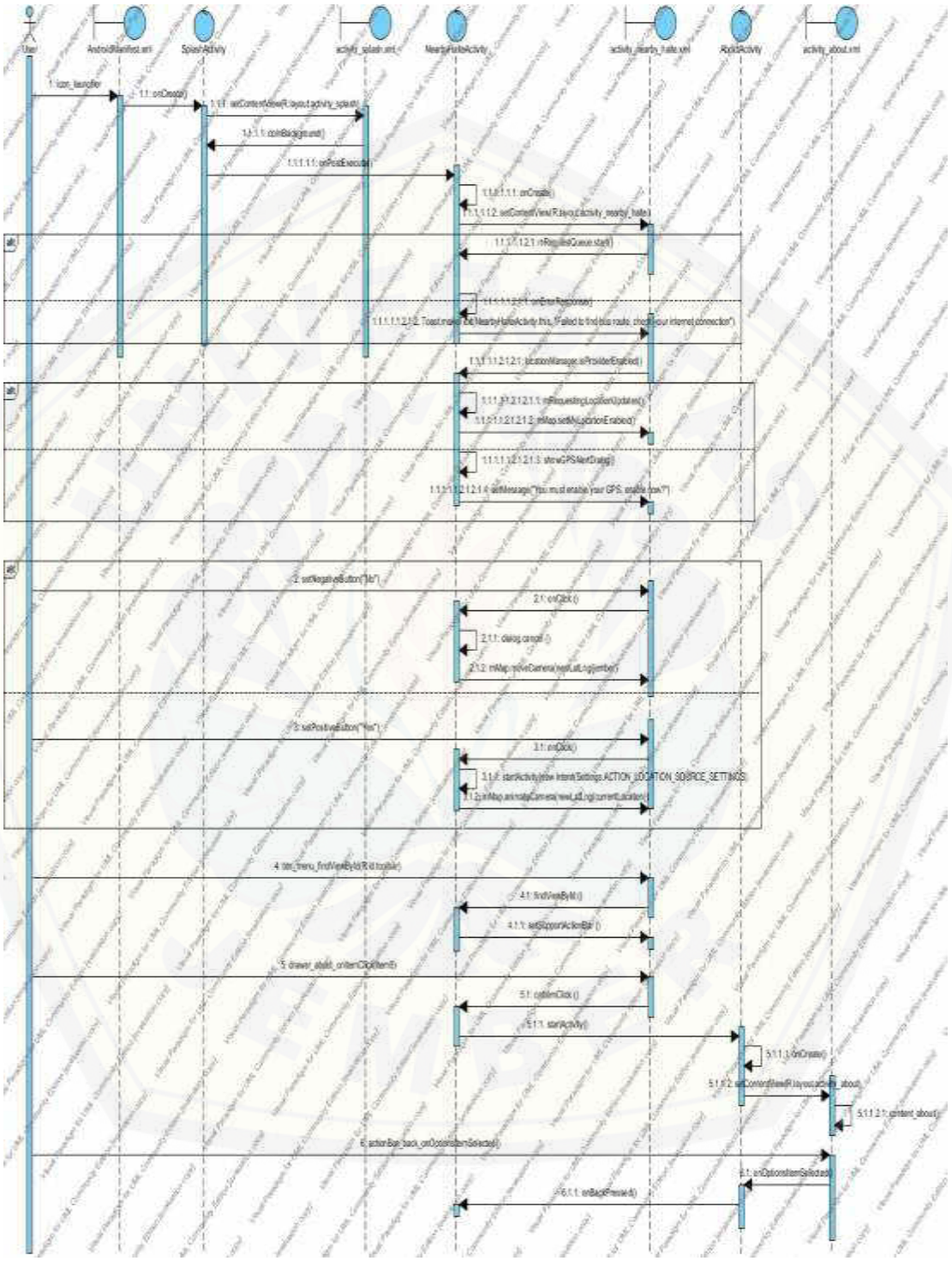
Sequence Diagram berfungsi untuk menggambarkan proses dan perubahan apa saja yang akan terjadi secara internal dan output apa yang dihasilkan pada aplikasi pencarian halte terdekat *bus* Trans SARBAGITA. *Sequence Diagram* dapat dilihat pada Gambar 4.5 sampai Gambar 4.7.



Gambar 4.5 *Sequence Diagram My Location*



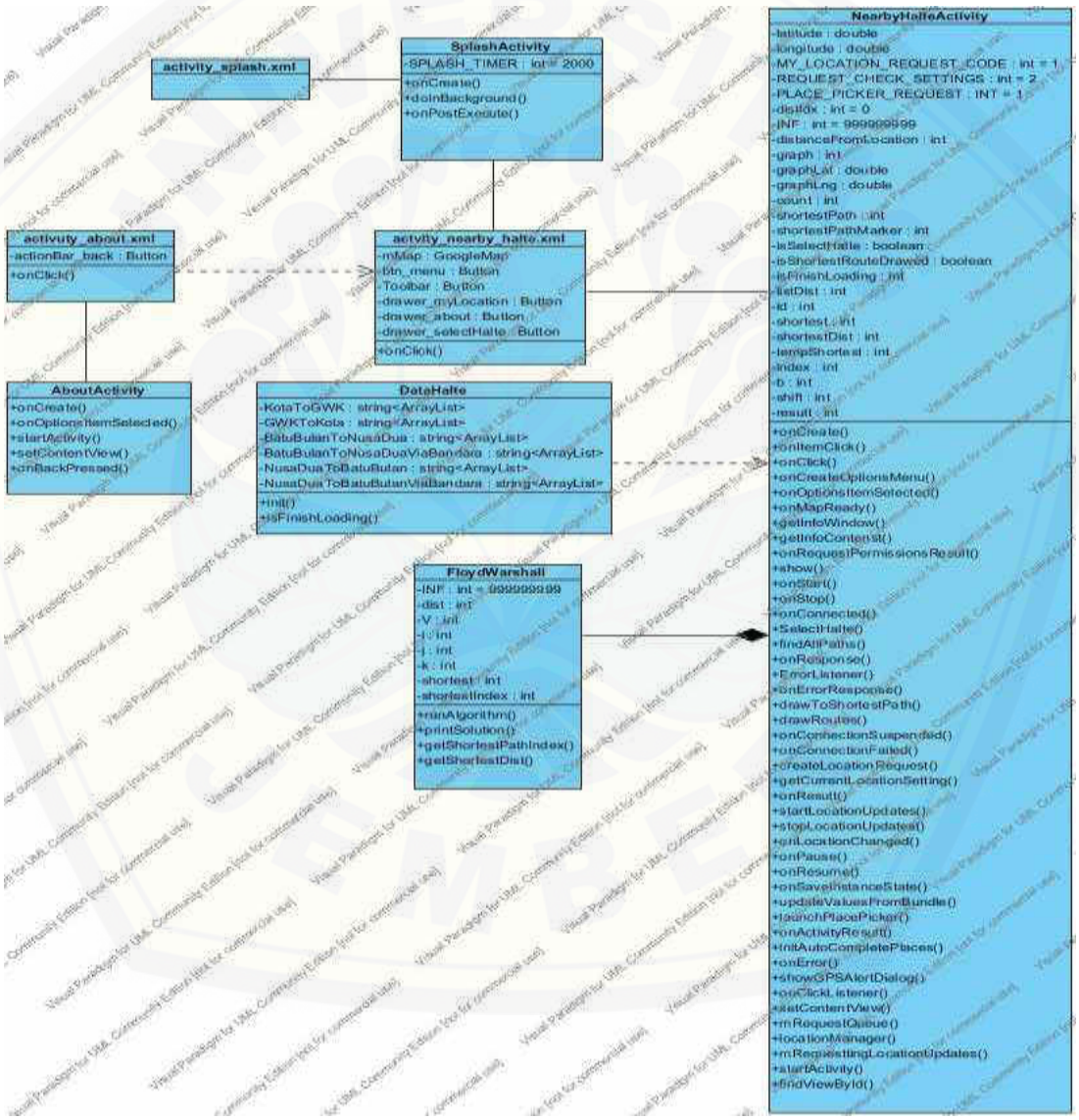
Gambar 4.6 *Sequence Diagram Bus Route*



Gambar 4.7 *Sequence Diagram About*

4.6 *Class Diagram*

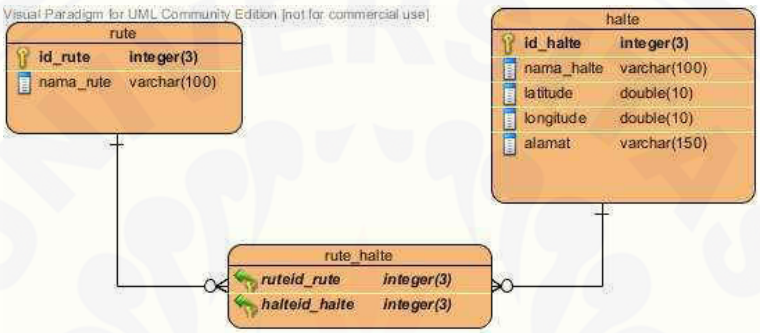
Class Diagram berfungsi untuk menggambarkan *class – class* atau objek apa saja yang akan digunakan untuk membuat aplikasi pencarian halte terdekat *bus* Trans SARBAGITA serta relasi atau hubungan yang terjadi pada *class – class* atau objek tersebut. *Class Diagram* aplikasi Pencarian Halte Terdekat dapat dlihat pada Gambar 4.8.



Gambar 4.8 *Class Diagram*

4.7 Entity Relation Diagram

ERD (Entity Relation Diagram) berfungsi untuk menggambarkan entitas - entitas apa saja yang akan digunakan pada aplikasi pencarian halte terdekat *bus* Trans SARBAGITA serta relasi atau hubungan yang terjadi pada entitas – entitas tersebut. *ERD (Entity Relation Diagram)* aplikasi pencarian halte terdekat *bus* Trans SARBAGITA dapat dilihat pada Gambar 4.9.



Gambar 4.9 *Entity Relationship Diagram*

4.8 Implementasi Perancangan

Setelah tahap desain perancangan selesai, tahap selanjutnya dalam penelitian ini yaitu tahap pengimplementasian. Tahap ini merupakan proses pengimplementasian desain perancangan yang telah dilakukan ke dalam bahasa pemograman. Bahasa pemrograman yang dipakai untuk proses pengaplikasian pada perangkat *mobile* android adalah bahasa pemrograman *Java* menggunakan *Android Studio* dan *database* menggunakan SQLite yang merupakan *embedded database* sehingga tidak memerlukan server untuk mengaksesnya.

4.9 Pengujian *White Box* dan *Black Box*

Tahap pengujian dilakukan untuk mengevaluasi kinerja dari sistem yang telah dibuat. Pengujian yang dilakukan dengan cara *White Box* dan *Black Box* . Pengujian *White Box* adalah pengujian dengan metode *test case* yang digambarkan berdasarkan notasi *diagram* alir. Pengujian *black box* adalah pengujian aspek *fundamental system* tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan

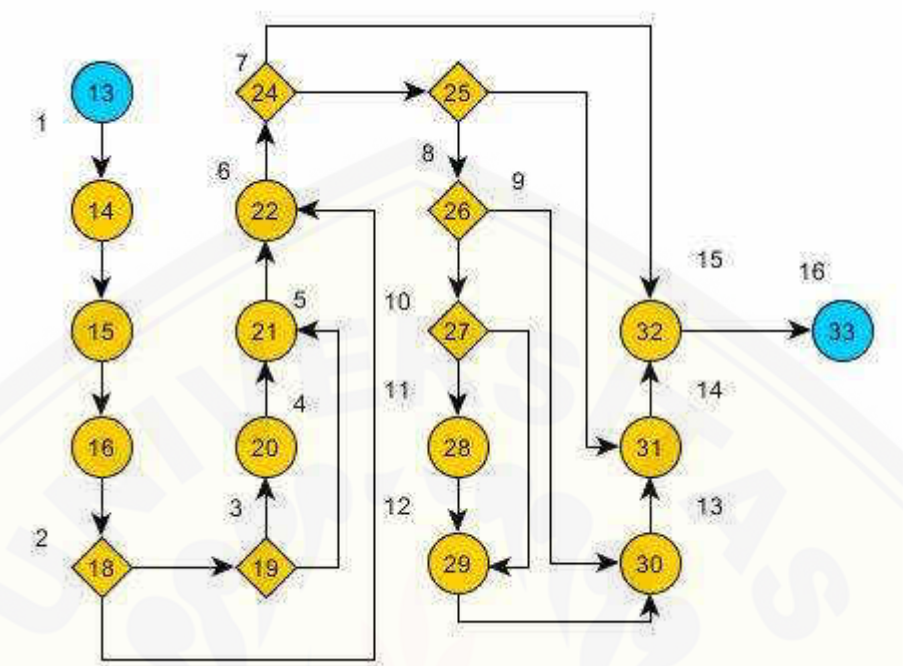
untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi sistem pencarian halte terdekat *bus* Trans SARBAGITA.

4.9.1 Pengujian White Box

Pengujian *white box* adalah tahap untuk menghitung kompleksitas sistem yang telah dibangun. Pengujian *white box* mengecek sistem berdasarkan alur perancangan yang telah dibuat sebelumnya. Pada penelitian ini metode yang digunakan dalam pengujian *white box* adalah *independent path* dengan menggunakan *cyclomatic complexity*. Pada pengujian ini akan membahas pengujian perhitungan algoritma *Floyd Warshall*. Listing program perhitungan algoritma *Floyd Warshall* dapat dilihat pada gambar 4.10 dan diagram alir perhitungan *cyclomatic complexity* pada algoritma *Floyd Warshall* dapat dilihat pada Gambar 4.10.

```
13      public void runAlgorithm(int graph[][]){
14          V = graph.length;
15          dist = new int[V][V];
16          int i, j, k;
17
18          for(i = 0; i < V; i++){
19              for(j = 0; j < V; j++){
20                  dist[i][j] = graph[i][j];
21              }
22          }
23
24          for(k = 0; k < V; k++){
25              for(i = 0; i < V; i++){
26                  for(j = 0; j < V; j++){
27                      if(dist[i][k] + dist[k][j] < dist[i][j]){
28                          dist[i][j] = dist[i][k] + dist[k][j];
29                      }
30                  }
31              }
32          }
33      }
```

Gambar 4.10 Listing Program Perhitungan Algoritma *Floyd Warshall*



CC = Edge – Node + 2

CC = 24 – 19 + 2 = 7

CC = 7

Gambar 4.11 *Diagram* Alir dan Perhitungan *Cyclomatic Complexity*

Jalur basis set pada pengujian program perhitungan algoritma *Floyd Warshall* adalah sebagai berikut :

Jalur 1 = 1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16

Jalur 2 = 1-2-6-7-8-9-10-11-12-13-14-15-16

Jalur 3 = 1-2-3-5-6-7-8-9-10-11-12-13-14-15-16

Jalur 4 = 1-2-3-4-5-6-7-15-16

Jalur 5 = 1-2-3-4-5-6-7-8-14-15-16

Jalur 6 = 1-2-3-4-5-6-7-8-9-13-14-15-16

Jalur 7 = 1-2-3-4-5-6-7-8-9-10-12-13-14-15-16

Pegujian kebenaran dari jalur yang telah terbentuk dapat dilihat pada tabel 4.6.

Tabel 4.6 Kebenaran Jalur Perhitungan *Floyd Warshall*

Test Case Method runAlgorithm ()	
Jalur 1	
Test Cast	Menghitung algoritma *Floyd Warshall*
Target yang diharapkan	Mendapatkan hasil lintasan terdekat
Hasil Pengujian	Benar
Path/Jalur	1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16
Jalur 2	
Test Cast	Mendapatkan titik awal (i)
Target yang diharapkan	Semua node menjadi titik awal (i)
Hasil Pengujian	Benar
Path/Jalur	1-2-6-7-8-9-10-11-12-13-14-15-16
Jalur 3	
Test Cast	Mendapatkan titik akhir (j)
Target yang diharapkan	Semua node menjadi titik akhir (j) dan membuat matrik awal (Weight Matrix)
Hasil Pengujian	Benar
Path/Jalur	1-2-3-5-6-7-8-9-10-11-12-13-14-15-16
Jalur 4	
Test Cast	Mendapatkan titik penghubung (k)
Target yang diharapkan	Semua node menjadi titik penghubung (k)
Hasil Pengujian	Benar
Path/Jalur	1-2-3-4-5-6-7-15-16
Jalur 5	
Test Cast	Iterasi pada titik awal (i)
Target yang diharapkan	Mendapatkan hasil iterasi (i)
Hasil Pengujian	Benar

Path/Jalur	1-2-3-4-5-6-7-8-14-15-16
Jalur 6	
Test Cast	Iterasi pada titik awal (j)
Target yang diharapkan	Mendapatkan hasil iterasi (j)
Hasil Pengujian	Benar
Path/Jalur	1-2-3-4-5-6-7-8-9-13-14-15-16
Jalur 7	
Test Cast	Melakukan perhitungan iterasi pada semua node
Target yang diharapkan	Mendapatkan hasil iterasi pada semua node dalam matrik hasil
Hasil Pengujian	Benar
Path/Jalur	1-2-3-4-5-6-7-8-9-10-12-13-14-15-16

4.9.2 Pengujian *Black-box*

Pengujian *black box* adalah pengujian aspek fundamental sistem tanpa memperhatikan struktur logika internal perangkat lunak. Metode ini digunakan untuk mengetahui apakah perangkat lunak berfungsi dengan benar. Pengujian *black box* merupakan metode perancangan data uji yang didasarkan pada spesifikasi perangkat lunak. Data uji dibangkitkan, dieksekusi pada perangkat lunak dan kemudian keluaran dari perangkat lunak dicek apakah telah sesuai dengan yang diharapkan. Pengujian *black box* dapat dilihat pada Tabel 4.7.

Tabel 4.7 Pengujian *Black Box* Aplikasi Bengkel Mobil Jember

No	Menu	Fungsi	Kasus	Hasil	Ket
1	*My Location*	Untuk menentukan posisi *user* saat	*User* menekan menu *my location*	Menampilkan posisi *user* saat menggunakan	OK

		menggunakan aplikasi		aplikasi berupa map	
2	*Bus Route*	Untuk mencari halte terdekat dari posisi *user* pada rute Kota – GWK	*User* menekan menu rute *bus* Kota – GWK	Menampilkan halte terdekat dari posisi *user* pada rute Kota – GWK	OK
		Untuk mencari halte terdekat dari posisi *user* pada rute GWK – Kota	*User* menekan menu rute *bus* GWK – Kota	Menampilkan halte terdekat dari posisi *user* pada rute GWK – Kota	OK
		Untuk mencari halte terdekat dari posisi *user* pada rute Batu Bulan – Nusa Dua	*User* menekan menu rute *bus* Batu Bulan – Nusa Dua	Menampilkan halte terdekat dari posisi *user* pada rute Batu Bulan – Nusa Dua	OK
		Untuk mencari halte terdekat dari posisi *user* pada rute Nusa Dua – Batu Bulan	*User* menekan menu rute *bus* Nusa Dua – Batu Bulan	Menampilkan halte terdekat dari posisi *user* pada rute Nusa Dua – Batu Bulan	OK
		Untuk mencari halte terdekat dari posisi *user* pada rute Batu Bulan – Nusa Dua Via Bandara	*User* menekan menu rute *bus* Batu Bulan – Nusa Dua Via Bandara	Menampilkan halte terdekat dari posisi *user* pada rute Batu Bulan – Nusa Dua Via Bandara	OK

		Untuk mencari halte terdekat dari posisi *user* pada rute Nusa Dua – Batu Bulan Via Bandara	*User* menekan menu rute *bus* Nusa Dua – Batu Bulan Via Bandara	Menampilkan halte terdekat dari posisi *user* pada rute Nusa Dua – Batu Bulan Via Bandara	OK
3	*About*	Untuk memberikan informasi tentang aplikasi Trans SARBAGITA *Bus Stop Finder*	*User* menekan menu *about*	Menampilkan informasi tentang aplikasi Trans SARBAGITA *Bus Stop Finder*	OK

BAB 6. PENUTUP

Bab ini merupakan bagian akhir dari penulisan skripsi yang berisi tentang kesimpulan dan saran. Kesimpulan merupakan hasil dari penelitian yang dilakukan dan saran digunakan untuk penelitian selanjutnya.

6.1 Kesimpulan

Kesimpulan dari sistem yang telah dibangun dalam penelitian ini adalah sebagai berikut :

1. Aplikasi pencarian halte terdekat *bus* Trans SARBAGITA sudah berhasil dibuat pada platform android dan dapat memberikan informasi halte terdekat dari posisi *user,* rute apa saja yang berada pada koridor 1 dan koridor 2, serta menampilkan pemetaan letak halte yang ada.
2. Algoritma *Floyd Warshall* diterapkan dalam perhitungan untuk menemukan halte terdekat dari posisi *user* dengan menggunakan parameter jarak dari titik koordinat *user* ke masing – masing titik koordinat halte yang ada. Setelah didapat jarak dari posisi *user* ke masing – masing halte kemudian dibandingkan dan dicari jarak yang paling kecil sebagai halte terdekat.
3. Menurut hasil perhitungan manual dengan aplikasi Trans SARBAGITA *Bus Stop Finder* didapatkan selisih hasil dari 50m – 150 m. Selisih hasil ini dikarenakan akurasi titik awal yang didapat oleh penulis dengan pencarian manual kurang tepat.
4. Dalam penelitian ini perhitungan iterasi untuk mencari jarak terpendek dari keempat dataset halte yang memiliki 12 node dilakukan sebanyak 12 kali. Iterasi dilakukan sebanyak 12 kali karena iterasi k = n (jumlah node).
5. Dalam penelitian ini hasil halte terdekat dari posisi user dengan titik koordinat latitude -8.635852 longitude 115.218144 pada koridor 2 adalah halte Batu Bulan dengan jarak 6350 m melalui rute A – B – C – D – E – F – G. Perhitungan manual

dengan aplikasi Trans SARBAGITA *Bus Stop Finder* menghasilkan halte terdekat Batu Bulan dari posisi *user*.

6. Proses pencarian halte terdekat menggunakan algoritma *Floyd Warshall* dalam aplikasi ini membutuhkan waktu proses yang relatif lama dikarenakan harus melakukan perhitungan dengan semua data halte yang ada dalam database. Setelah perhitungan selesai, semua jarak dari posisi *user* ke masing – masing halte dibandingkan kemudian dicari yang memiliki jarak terkecil.
7. Melakukan pencarian halte terdekat dengan menggunakan aplikasi Trans SARBAGITA *bus stop finder* menjadi lebih efisien daripada menggunakan Google Maps karena langsung menampilkan halte terdekat dan rute yang terdekat menuju halte.

6.2 Saran

Hasil dari penelitian ini belum sempurna, oleh karena itu untuk meningkatkan hasil yang dicapai beberapa saran untuk mengembangkan penelitian ini :

1. Pada penelitian ini penulis hanya menggunakan 2 koridor *bus* Trans SARBAGITA saja, disarankan pada penelitian berikutnya agar menambahkan koridor – koridor lainnya.
2. Pada penelitian ini hasil rute terpendek menuju halte hanya berupa *marker* dan *polyline* berupa lintasan saja, penulis menyarankan agar ada navigasi atau penjelasan jalan menuju halte terdekat untuk mempermudah *user*.
3. Aplikasi perlu dikembangkan dengan menambahkan parameter lain yang mempengaruhi pencarian halte terdekat, misalnya waktu tempuh perjalanan.
4. Perlu dilakukan pengembangan pada sistem operasi mobile yang lainnya seperti Windows Phone dan iOS.

DAFTAR PUSTAKA

Ardiansyah I., Hakim D.K. (2012). Rancang Bangun Aplikasi untuk Menentukan Jalur Terpendek Menggunakan Algoritma Floyd Warshall di Lokasi Wisata Purbalingga. *JUITA ISSN: 2086-9398 Vol. II Nomor 2*, 133-143.

Gunawan, H. (2013). Rancang Bangun Aplikasi Travel Guide Banyumas Berbasis Android. *(Doctoral dissertation, UAJY)*, http://e-journal.uajy.ac.id/.

Ipulhe. (2015, 6 7). *Pengertian Aplikasi*. Dipetik 1 15, 2016, dari http://www.ipulhe.com: http://www.ipulhe.com/pengertian-aplikasi/

Istyanto, M. H. (2014). *Rancang Bangun Aplikasi Pencarian Jalur Terpendek Menggunakan Algoritma Floyd Warshall (Studi Kasus Kota Singkawang).* Pontianak: Universitas Tanjungpura.

Kriswanto R.Y., Bendi R.K.J. (2014). Penentuan Jarak Terpendek Rute Transmusi dengan Algoritma Floyd-Warshall. *SEMINAR NASIONAL TEKNOLOGI INFORMASI & KOMUNIKASI TERAPAN 2014(SEMANTIK 2014)*, 209-216.

Liwang, R. (2013). *Rancang Bangun Aplikasi Pencarian Rute Terpendek Tempat Wisata Dengan Memanfaatkan Google Maps API (Studi Kasus : Kabupaten Kulon Progo).* Yogyakarta: Universitas Atma Jaya.

Prima E., Rahanda J.I. (2013). *TUGAS 3 TUGAS KELOMPOK MATAKULIAH SISTEM ANGKUTAN UMUM TSI – 462 Tentang PERENCANAAN HALTE / BUS STOP BUS RAPID TRANSIT (BRT) / TRANS PADANG DI KOTA PADANG*. Dipetik 5 16, 2016, dari https://www.academia.edu: https://www.academia.edu/5225356/TUGAS_3_TUGAS_KELOMPOK_MATAKULIAH_SISTEM_ANGKUTAN_UMUM_TSI_462_Tentang_PERENCANAAN_HALTE_BUS_STOP_BUS_RAPID_TRANSIT_BRT_TRANS_PADANG_DI_KOTA_PADANG

Safaat N, H. (2012). *Buku Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android.* Bandung: INFORMATIKA.

Setyawan, D. (2012, agustus). *Definisi Aplikasi*. Dipetik Januari 15, 2016, dari http://definisimu.blogspot.co.id: http://definisimu.blogspot.co.id/2012/08/definisi-aplikasi.html

Sihotang, D. E. (2015). *PENGARUH STATUS SOSIAL EKONOMI MASYARAKAT KOTA DENPASAR TERHADAP PEMILIHAN TRANS SARBAGITA SEBAGAI MODA TRANSPORTASI (STUDI di KORIDOR I dan II)*. Bali: Universitas Udayana.

Siswanto. (2012). *SISTEM INFORMASI GEOGRAFIS OBJEK WISATA MENGGUNAKAN GOOGLE MAPS API*. Surabaya: Politeknik Elektronika Negeri Surabaya.

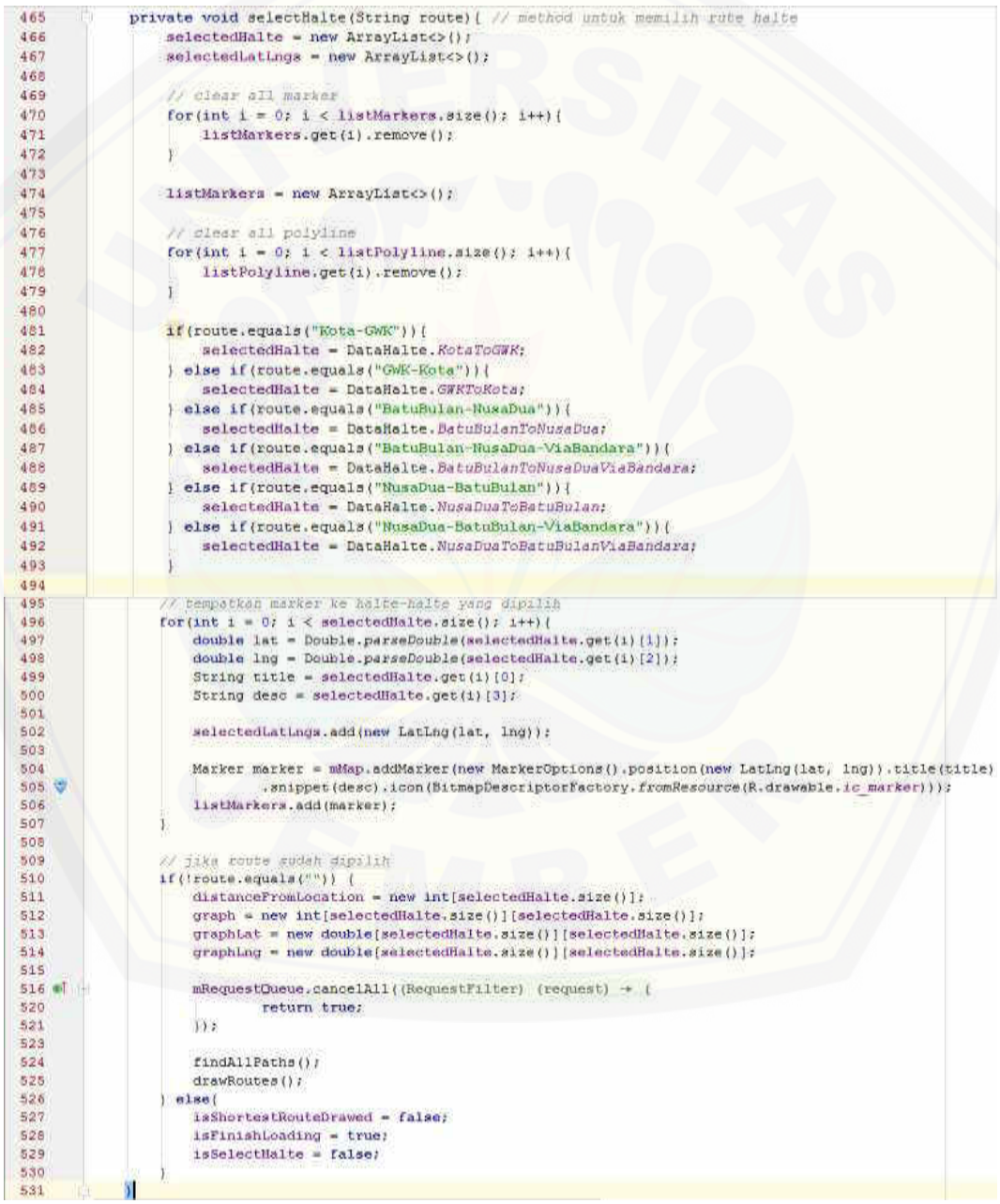
Wiwaha, A. (2013, January 31). *Tinjauan Transportasi Secara Umum*. Dipetik 4 16, 2016, dari http://studyandlearningnow.blogspot.co.id: http://studyandlearningnow.blogspot.co.id/2013/01/tinjauan-transportasi-secara-umum_31.html

Yusaputra, R. (2013). *Aplikasi Mobile Pencarian Rute Terpendek Lokasi Fasilitas Umum Berbasis Android Menggunakan Algoritma Floyd Warshall*. Riau: Universitas Islam Negeri Sultan Syarif Kasim.

LAMPIRAN

LAMPIRAN A. PENULISAN KODE PROGRAM

A1. Penulisan Kode Program Memilih Rute Halte



```
465     private void selectHalte(String route){ // method untuk memilih rute halte
466         selectedHalte = new ArrayList<>();
467         selectedLatLngs = new ArrayList<>();
468
469         // clear all marker
470         for(int i = 0; i < listMarkers.size(); i++){
471             listMarkers.get(i).remove();
472         }
473
474         listMarkers = new ArrayList<>();
475
476         // clear all polyline
477         for(int i = 0; i < listPolyline.size(); i++){
478             listPolyline.get(i).remove();
479         }
480
481         if(route.equals("Kota-GWK")){
482             selectedHalte = DataHalte.KotaToGWK;
483         } else if(route.equals("GWK-Kota")){
484             selectedHalte = DataHalte.GWKToKota;
485         } else if(route.equals("BatuBulan-NusaDua")){
486             selectedHalte = DataHalte.BatuBulanToNusaDua;
487         } else if(route.equals("BatuBulan-NusaDua-ViaBandara")){
488             selectedHalte = DataHalte.BatuBulanToNusaDuaViaBandara;
489         } else if(route.equals("NusaDua-BatuBulan")){
490             selectedHalte = DataHalte.NusaDuaToBatuBulan;
491         } else if(route.equals("NusaDua-BatuBulan-ViaBandara")){
492             selectedHalte = DataHalte.NusaDuaToBatuBulanViaBandara;
493         }
494
495         // tempatkan marker ke halte-halte yang dipilih
496         for(int i = 0; i < selectedHalte.size(); i++){
497             double lat = Double.parseDouble(selectedHalte.get(i)[1]);
498             double lng = Double.parseDouble(selectedHalte.get(i)[2]);
499             String title = selectedHalte.get(i)[0];
500             String desc = selectedHalte.get(i)[3];
501
502             selectedLatLngs.add(new LatLng(lat, lng));
503
504             Marker marker = mMap.addMarker(new MarkerOptions().position(new LatLng(lat, lng)).title(title)
505                     .snippet(desc).icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_marker)));
506             listMarkers.add(marker);
507         }
508
509         // jika route sudah dipilih
510         if(!route.equals("")) {
511             distanceFromLocation = new int[selectedHalte.size()];
512             graph = new int[selectedHalte.size()][selectedHalte.size()];
513             graphLat = new double[selectedHalte.size()][selectedHalte.size()];
514             graphLng = new double[selectedHalte.size()][selectedHalte.size()];
515
516             mRequestQueue.cancelAll((RequestFilter) (request) -> {
520                     return true;
521             });
523
524             findAllPaths();
525             drawRoutes();
526         } else{
527             isShortestRouteDrawed = false;
528             isFinishLoading = true;
529             isSelectHalte = false;
530         }
531     }
```

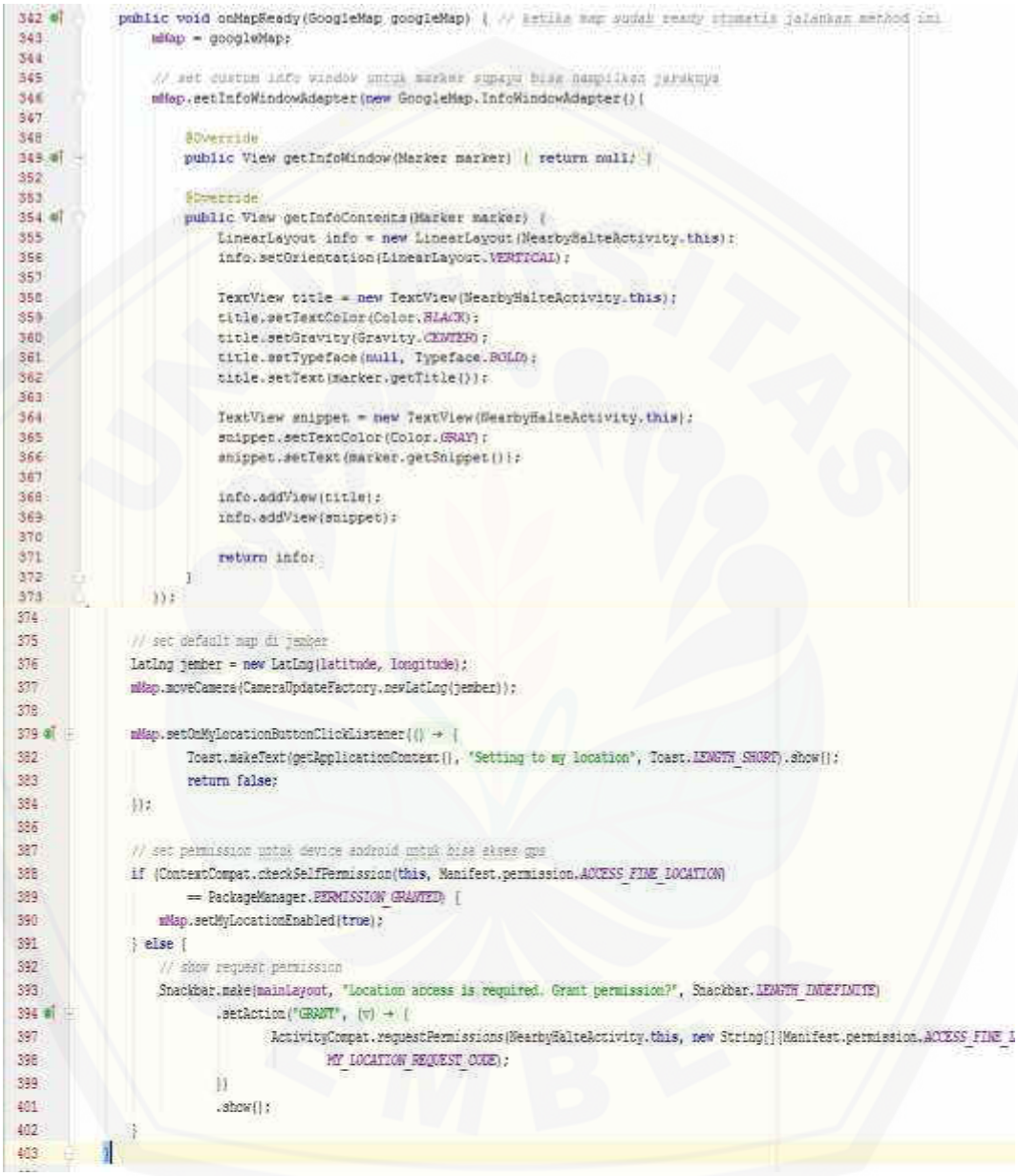
A2. Penulisan Kode Program *Activity* Memilih Rute Halte

```
public void onClick(View view) {
    if(isSelectHalte){
        if(isFinishLoading) {
            if (!isShortestRouteDrawed) {
                // draw shortest route
                drawToShortestPath(graphLat[0][shortestPath], graphLng[0][shortestPath]);
            }
            listMarkers.get(shortestPathMarker).showInfoWindow();
            mMap.animateCamera(CameraUpdateFactory.newLatLng(new LatLng(graphLat[0][shortestPath], graphLng[0][shortestPath])));
            Log.e("shortest path: ", "" + shortestPath);
        } else {
            Toast.makeText(NearbyHalteActivity.this, "Please wait until all halte loaded", Toast.LENGTH_LONG).show();
        }
    } else {
        Snackbar.make(view, "Select bus route first", Snackbar.LENGTH_LONG)
                .setAction("Select", (v) -> {
                    drawer.openDrawer();
                }).show();
    }
}
```

A3. Penulisan Kode Program *Decode* Titik *Polyline*

```
private List<LatLng> decodePoly(String encoded) {

    List<LatLng> poly = new ArrayList<>();
    int index = 0, len = encoded.length();
    int lat = 0, lng = 0;

    while (index < len) {
        int b, shift = 0, result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlat = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lat += dlat;

        shift = 0;
        result = 0;
        do {
            b = encoded.charAt(index++) - 63;
            result |= (b & 0x1f) << shift;
            shift += 5;
        } while (b >= 0x20);
        int dlng = ((result & 1) != 0 ? ~(result >> 1) : (result >> 1));
        lng += dlng;

        LatLng p = new LatLng((((double) lat / 1E5)),
                (((double) lng / 1E5)));
        poly.add(p);
    }

    return poly;
}
```

A4. Penulisan Kode Program Map Siap Digunakan



```
public void onMapReady(GoogleMap googleMap) { // ketika map sudah ready otomatis jalankan method ini
    mMap = googleMap;

    // set custom info window untuk marker supaya bisa menampilkan jaraknya
    mMap.setInfoWindowAdapter(new GoogleMap.InfoWindowAdapter(){

        @Override
        public View getInfoWindow(Marker marker) { return null; }

        @Override
        public View getInfoContents(Marker marker) {
            LinearLayout info = new LinearLayout(NearbyHalteActivity.this);
            info.setOrientation(LinearLayout.VERTICAL);

            TextView title = new TextView(NearbyHalteActivity.this);
            title.setTextColor(Color.BLACK);
            title.setGravity(Gravity.CENTER);
            title.setTypeface(null, Typeface.BOLD);
            title.setText(marker.getTitle());

            TextView snippet = new TextView(NearbyHalteActivity.this);
            snippet.setTextColor(Color.GRAY);
            snippet.setText(marker.getSnippet());

            info.addView(title);
            info.addView(snippet);

            return info;
        }
    });

    // set default map di jember
    LatLng jember = new LatLng(latitude, longitude);
    mMap.moveCamera(CameraUpdateFactory.newLatLng(jember));

    mMap.setOnMyLocationButtonClickListener(() -> {
            Toast.makeText(getApplicationContext(), "Setting to my location", Toast.LENGTH_SHORT).show();
            return false;
    });

    // set permission untuk device android untuk bisa akses gps
    if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION)
            == PackageManager.PERMISSION_GRANTED) {
        mMap.setMyLocationEnabled(true);
    } else {
        // show request permission
        Snackbar.make(mainLayout, "Location access is required. Grant permission?", Snackbar.LENGTH_INDEFINITE)
                .setAction("GRANT", (v) -> {
                        ActivityCompat.requestPermissions(NearbyHalteActivity.this, new String[]{Manifest.permission.ACCESS_FINE_L
                                MY_LOCATION_REQUEST_CODE);
                })
                .show();
    }
}
```

A5. Penulisan Kode Program Koneksi Google *Maps*

```
public void onConnected(Bundle connectionHint) { // jika google api client sudah connect
    if (ActivityCompat.checkSelfPermission(this, Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        return;
    }
    // set ke lokasi terakhir / sekarang berdasarkan GPS
    mLastLocation = LocationServices.FusedLocationApi.getLastLocation(mGoogleApiClient);
    if (mLastLocation != null) {
        latitude = mLastLocation.getLatitude();
        longitude = mLastLocation.getLongitude();
        mCurrentLocation = mLastLocation;

        createLocationRequest();
    }

    if(mRequestingLocationUpdates){
//            startLocationUpdates();
    }

    // init auto complete places
    initAutocompletePlaces();
}
```

A6. Penulisan Kode Program *Setting* Lokasi Sekarang

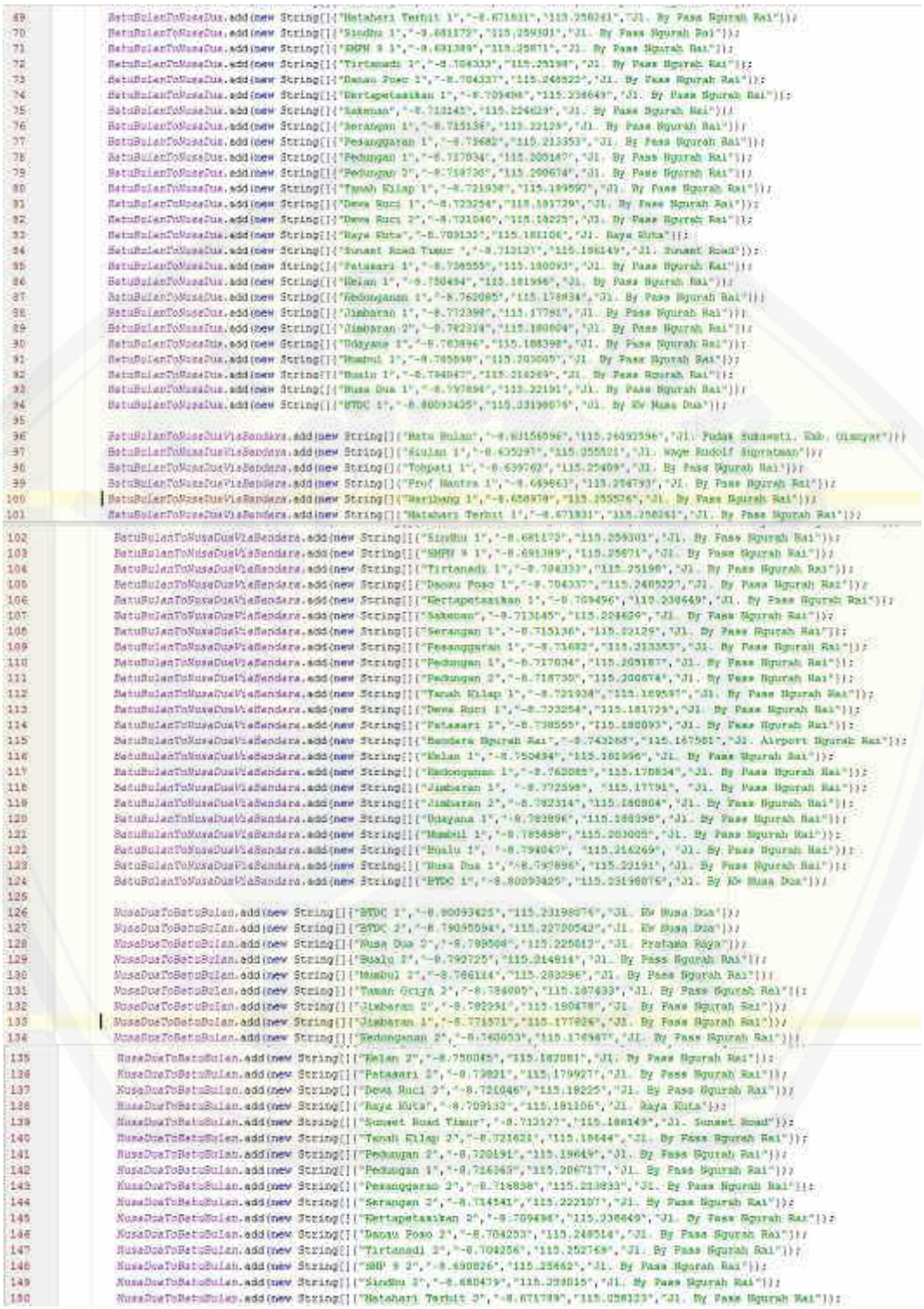
```
private void getCurrentLocationSettings() {
    LocationSettingsRequest.Builder builder = new LocationSettingsRequest.Builder()
            .addLocationRequest(mLocationRequest);

    PendingResult<LocationSettingsResult> result =
            LocationServices.SettingsApi.checkLocationSettings(mGoogleApiClient, builder.build());

    result.setResultCallback((ResultCallback) (result) -> {
            final Status status = result.getStatus();
            final LocationSettingsStates states = result.getLocationSettingsStates();
            switch (status.getStatusCode()) {
                case LocationSettingsStatusCodes.SUCCESS:

                    LatLng currentLocation = new LatLng(latitude, longitude);
                    mMap.animateCamera(CameraUpdateFactory.newLatLng(currentLocation));

                    break;
                case LocationSettingsStatusCodes.RESOLUTION_REQUIRED:
                    try {
                        status.startResolutionForResult(NearbyHalteActivity.this, REQUEST_CHECK_SETTINGS);
                    } catch (IntentSender.SendIntentException e) {
                    }
                    break;
                case LocationSettingsStatusCodes.SETTINGS_CHANGE_UNAVAILABLE:
                    break;
            }
        });
}
```

A7. Penulisan Kode Program *GPS Alert Dialog*

```
1013        private void showGPSAlertDialog(){
1014            final AlertDialog.Builder builder = new AlertDialog.Builder(this);
1015            builder.setMessage("You must enable your GPS, enable now?")
1016                    .setCancelable(false)
1017                    .setPositiveButton("Yes", (dialog, which) -> {
1020                            startActivity(new Intent(Settings.ACTION_LOCATION_SOURCE_SETTINGS));
1021                    })
1023                    .setNegativeButton("No", (dialog, which) -> {
1026                            dialog.cancel();
1027                    });
1029
1030            final AlertDialog dialog = builder.create();
1031            dialog.show();
1032        }
```

A8. Penulisan Kode Program *Class Helper* Data Halte

```
7   public class DataHalte {
8       public static List<String[]> KotaToGWK = new ArrayList<>();
9       public static List<String[]> GWKToKota = new ArrayList<>();
10
11      public static List<String[]> BatuBulanToNusaDua = new ArrayList<>();
12      public static List<String[]> BatuBulanToNusaDuaViaBandara = new ArrayList<>();
13
14      public static List<String[]> NusaDuaToBatuBulan = new ArrayList<>();
15      public static List<String[]> NusaDuaToBatuBulanViaBandara = new ArrayList<>();
16
17      public static void init(){
18          KotaToGWK.add(new String[]{"Kamboja","-8.651216","115.224456","Jl. Kamboja"});
19          KotaToGWK.add(new String[]{"Surapati 1","-8.656535","115.230728","Jl. Surapati"});
20          KotaToGWK.add(new String[]{"Sudirman 1","-8.666101","115.217993","Jl. PB Sudirman"});
21          KotaToGWK.add(new String[]{"Sudirman 2","-8.671727","115.21822","Jl. PB Sudirman"});
22          KotaToGWK.add(new String[]{"Sudirman 3","-8.675565","115.217754","Jl. PB Sudirman"});
23          KotaToGWK.add(new String[]{"Pesanggaran 1","-8.71682","115.213353","Jl. By Pass Ngurah Rai"});
24          KotaToGWK.add(new String[]{"Pedungan 1","-8.717034","115.205167","Jl. By Pass Ngurah Rai"});
25          KotaToGWK.add(new String[]{"Pedungan 2","-8.718138","115.200474","Jl. By Pass Ngurah Rai"});
26          KotaToGWK.add(new String[]{"Tanah Kilap 1","-8.721938","115.199597","Jl. By Pass Ngurah Rai"});
27          KotaToGWK.add(new String[]{"Dewa Ruci 1","-8.723254","115.191729","Jl. By Pass Ngurah Rai"});
28          KotaToGWK.add(new String[]{"Patasari 1","-8.726555","115.190093","Jl. By Pass Ngurah Rai"});
29          KotaToGWK.add(new String[]{"Selan 1","-8.750494","115.181896","Jl. By Pass Ngurah Rai"});
30          KotaToGWK.add(new String[]{"Kedonganan 1","-8.762805","115.179834","Jl. By Pass Ngurah Rai"});
31          KotaToGWK.add(new String[]{"Jimbaran 1","-8.772596","115.17791","Jl. By Pass Ngurah Rai"});
32          KotaToGWK.add(new String[]{"Udayana 1","-8.793035","115.176713","Jl. Raya Kampus Unud, Kuta Selatan"});
33          KotaToGWK.add(new String[]{"Pertanian","-8.792801","115.17697","Jl. Raya Kampus Unud, Kuta Selatan"});
34          KotaToGWK.add(new String[]{"Udayana 3","-8.795735","115.176566","Jl. Raya Kampus Unud, Kuta Selatan"});
35          KotaToGWK.add(new String[]{"Rektorat udayana","-8.797471","115.171659","Jl. Raya Kampus Unud, Kuta Selatan"});
36          KotaToGWK.add(new String[]{"Udayana 4","-8.79952","115.170554","Jl. Raya Kampus Unud, Kuta Selatan"});
37          KotaToGWK.add(new String[]{"Ekonomi","-8.80061","115.169659","Jl. Raya Kampus Unud, Kuta Selatan"});
38          KotaToGWK.add(new String[]{"Udayana 5","-8.799355","115.168324","Jl. Raya Kampus Unud, Kuta Selatan"});
39          KotaToGWK.add(new String[]{"GWK","-8.808114","115.16959","Jl. Kv Garuda Wisnu Kencana, Kuta Selatan"});
40
41          GWKToKota.add(new String[]{"GWK","-8.808114","115.16959","Jl. Kv Garuda Wisnu Kencana, Kuta Selatan"});
42          GWKToKota.add(new String[]{"Bekas Kari","-8.80380","115.169405","Jl. Ciyeah, Kuta Selatan"});
43          GWKToKota.add(new String[]{"Udayana 5","-8.799417","115.168317","Jl. Raya Kampus Unud, Kuta Selatan"});
44          GWKToKota.add(new String[]{"Ekonomi","-8.800018","115.169658","Jl. Raya Kampus Unud, Kuta Selatan"});
45          GWKToKota.add(new String[]{"Udayana 4","-8.799485","115.170998","Jl. Raya Kampus Unud, Kuta Selatan"});
46          GWKToKota.add(new String[]{"Udayana 3","-8.795222","115.17577","Jl. Raya Kampus Unud, Kuta Selatan"});
47          GWKToKota.add(new String[]{"Pertanian","-8.792801","115.17697","Jl. Raya Kampus Unud, Kuta Selatan"});
48          GWKToKota.add(new String[]{"Udayana 2","-8.788563","115.177411","Jl. Raya Kampus Unud, Kuta Selatan"});
49          GWKToKota.add(new String[]{"Udayana 1","-8.782555","115.178558","Jl. Raya Kampus Unud, Kuta Selatan"});
50          GWKToKota.add(new String[]{"Jimbaran 1","-8.771571","115.177829","Jl. By Pass Ngurah Rai"});
51          GWKToKota.add(new String[]{"Kedonganan 2","-8.765053","115.178047","Jl. By Pass Ngurah Rai"});
52          GWKToKota.add(new String[]{"Selan 2","-8.750945","115.182051","Jl. By Pass Ngurah Rai"});
53          GWKToKota.add(new String[]{"Patasari 2","-8.73851","115.179827","Jl. By Pass Ngurah Rai"});
54          GWKToKota.add(new String[]{"Tanah Kilap 2","-8.731691","115.19644","Jl. By Pass Ngurah Rai"});
55          GWKToKota.add(new String[]{"Pedungan 2","-8.728195","115.196469","Jl. By Pass Ngurah Rai"});
56          GWKToKota.add(new String[]{"Pedungan 1","-8.718363","115.204719","Jl. By Pass Ngurah Rai"});
57          GWKToKota.add(new String[]{"Pesanggaran 2","-8.716856","115.213833","Jl. By Pass Ngurah Rai"});
58          GWKToKota.add(new String[]{"RSUP Sanglah","-8.67562","115.215327","Jl. Diponegoro "});
59          GWKToKota.add(new String[]{"Sudirman 2","-8.672726","115.217456","Jl. PB Sudirman"});
60          GWKToKota.add(new String[]{"Surapati 2","-8.654859","115.218936","Jl. Surapati "});
61          GWKToKota.add(new String[]{"Kamboja 2","-8.65236","115.224474","Jl. Kamboja"});
62          GWKToKota.add(new String[]{"Kamboja","-8.651216","115.224456","Jl. Kamboja"});
63
64          BatuBulanToNusaDua.add(new String[]{"Batu Bulan","-8.63156546","115.24892596","Jl. Pura Subak Sari, Kec. Gianyar"});
65          BatuBulanToNusaDua.add(new String[]{"Sinlan 1","-8.635297","115.255531","Jl. Wage Rudolf Supratman"});
66          BatuBulanToNusaDua.add(new String[]{"Tutpati 1","-8.639767","115.25409","Jl. By Pass Ngurah Rai"});
67          BatuBulanToNusaDua.add(new String[]{"Prof Mantra 1","-8.643963","115.254743","Jl. By Pass Ngurah Rai"});
68          BatuBulanToNusaDua.add(new String[]{"Heriberg 1","-8.650975","115.2535767","Jl. By Pass Ngurah Rai"});
```

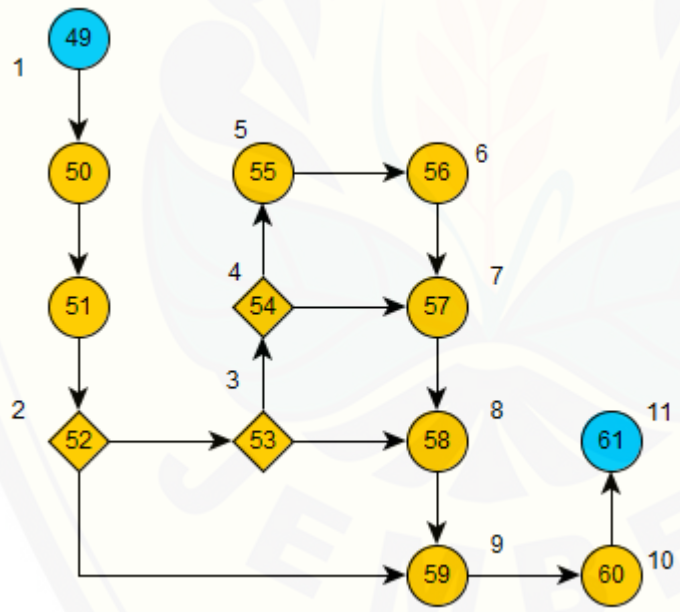


```java
        NusaDuaToBatuBulan.add(new String[]{"Waribang 2","-8.655938","115.253798","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulan.add(new String[]{"Prof IB. Mantra 2","-8.650495","115.254561","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulan.add(new String[]{"Tohpati 2","-8.640958","115.25447","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulan.add(new String[]{"Siulan 2","-8.633764","115.254948","Jl. Wage Rudolf Supratman"});
        NusaDuaToBatuBulan.add(new String[]{"Batu Bulan","-8.63156598","115.26092598","Jl. Pudak, Sukawati Kab. Gianyar"});

        NusaDuaToBatuBulanViaBandara.add(new String[]{"BTDC 1","-8.80093425","115.23196076","Jl. Kw Nusa Dua"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"BTDC 2","-8.79095594","115.22720542","Jl. Kw Nusa Dua"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Nusa Dua 2","-8.789508","115.225612","Jl. Pratama Raya"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Bualu 2","-8.792709","115.214814","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Mumbul 2","-8.786114","115.202296","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Taman Griya 2","-8.784005","115.187433","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Jimbaran 2","-8.782291","115.180478","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Jimbaran 1","-8.771571","115.177826","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Kedonganan 2","-8.760053","115.178947","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Kelan 2","-8.758045","115.182081","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Bandara Ngurah Rai","-8.743260","115.167581","Jl. Airport Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Patasari 2","-8.73921","115.179927","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Tanah Kilap 2","-8.721621","115.18644","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Pedungan 2","-8.720191","115.19649","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Pedungan 1","-8.718367","115.204757","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Pesanggaran 2","-8.718858","115.213833","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Serangan 2","-8.714541","115.222107","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Kertapetasikan 2","-8.709496","115.235649","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Danau Poso 2","-8.704253","115.246514","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Tirtanadi 2","-8.704254","115.252769","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"SMP 8 2","-8.698026","115.25862","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Sindhu 2","-8.680470","115.259015","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Matahari Terbit 2","-8.671789","115.260133","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Waribang 2","-8.655938","115.253798","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Prof IB. Mantra 2","-8.650495","115.254561","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Tohpati 2","-8.640958","115.25447","Jl. By Pass Ngurah Rai"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Siulan 2","-8.633764","115.254948","Jl. Wage Rudolf Supratman"});
        NusaDuaToBatuBulanViaBandara.add(new String[]{"Batu Bulan","-8.63156598","115.26092598","Jl. Pudak, Sukawati Kab. Gianyar"});
    }
}
```

LAMPIRAN B. PENGUJIAN *WHITE BOX*

B1. Pengujian *White Box* Mendapatkan *Shortest Path*

```
49      public int getShorthestPathIndex() {
50          int shortest = INF;
51          int shortestIndex = 1;
52          for (int i = 0; i < V; i++) {
53              for (int j = 0; j < V; j++) {
54                  if (shortest > dist[i][j]) {
55                      shortest = dist[i][j];
56                      shortestIndex = j;
57                  }
58              }
59          }
60          return shortestIndex;
61      }
```



CC = Edge – Node + 2

CC = 15 – 13 + 2 = 4

CC = 4

LAMPIRAN C. DATA HALTE

Koridor 1			
Kota – GWK			
Kamboja	-8.651216	115.224456	Jl. Kamboja
Surapati 1	-8.656533	115.220728	Jl. Surapati
Sudirman 1	-8.666101	115.217993	Jl. PB Sudirman
Sudirman 2	-8.671727	115.21822	Jl. PB Sudirman
Sudirman 3	-8.675565	115.217754	Jl. PB Sudirman
Pesanggaran 1	-8.71682	115.213353	Jl. By Pass Ngurah Rai
Pedungan 1	-8.717034	115.205187	Jl. By Pass Ngurah Rai
Pedungan 2	-8.718730	115.200674	Jl. By Pass Ngurah Rai
Tanah Kilap 1	-8.721938	115.189597	Jl. By Pass Ngurah Rai
Dewa Ruci 1	-8.723254	115.181729	Jl. By Pass Ngurah Rai
Patasari 1	-8.738555	115.180093	Jl. By Pass Ngurah Rai
Kelan 1	-8.750494	115.181996	Jl. By Pass Ngurah Rai
Kedonganan 1	-8.762085	115.178834	Jl. By Pass Ngurah Rai
Jimbaran 1	-8.772398	115.17791	Jl. By Pass Ngurah Rai
Udayana 1	-8.783035	115.178713	Jl. Raya Kampus Unud, Kuta Selatan
Pertanian	-8.792801	115.17697	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 3	-8.795735	115.175566	Jl. Raya Kampus Unud, Kuta Selatan
Rektorat udayana	-8.797471	115.171659	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 4	-8.79952	115.170954	Jl. Raya Kampus Unud, Kuta Selatan
Ekonomi	-8.80001	115.169575	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 5	-8.798333	115.161824	Jl. Raya Kampus Unud, Kuta Selatan
GWK	-8.806114	115.15919	Jl. Kw Garuda Wisnu Kencana, Kuta Selatan
GWK – KOTA			
GWK	-8.806114	115.15919	Jl. Kw Garuda Wisnu Kencana, Kuta Selatan
Mekar Sari	-8.803807	115.159605	Jl. Uluwatu, Kuta Selatan
Udayana 5	-8.798617	115.162717	Jl. Raya Kampus Unud, Kuta Selatan

Ekonomi	-8.800018	115.169638	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 4	-8.798445	115.170996	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 3	-8.795222	115.17577	Jl. Raya Kampus Unud, Kuta Selatan
Pertanian	-8.792801	115.17697	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 2	-8.788563	115.177111	Jl. Raya Kampus Unud, Kuta Selatan
Udayana 1	-8.782535	115.178655	Jl. Raya Kampus Unud, Kuta Selatan
Jimbaran 1	-8.771571	115.177826	Jl. By Pass Ngurah Rai
Kedonganan 2	-8.760053	115.178947	Jl. By Pass Ngurah Rai
Kelan 2	-8.750045	115.182081	Jl. By Pass Ngurah Rai
Patasari 2	-8.73821	115.179927	Jl. By Pass Ngurah Rai
Tanah Kilap 2	-8.721621	115.18644	Jl. By Pass Ngurah Rai
Pedungan 2	-8.720191	115.19649	Jl. By Pass Ngurah Rai
Pedungan 1	-8.716363	115.206717	Jl. By Pass Ngurah Rai
Pesanggaran 2	-8.716838	115.213833	Jl. By Pass Ngurah Rai
RSUP Sanglah	-8.67562	115.215327	Jl. Diponegoro
Sudirman 2	-8.672726	115.217956	Jl. PB Sudirman
Surapati 2	-8.65629	115.218996	Jl. Surapati
Kamboja 2	-8.652361	115.224474	Jl. Kamboja
Kamboja	-8.651216	115.224456	Jl. Kamboja

Koridor 2			
Halte Batu Bulan – Nusa Dua			
Nama Halte	Latitude	Longitude	Alamat
Batu Bulan	-8.63156596	115.26092596	Jl. Pudak Sukawati, Kab. Gianyar
Siulan 1	-8.635297	115.255521	Jl. Wage Rudolf Supratman
Tohpati 1	-8.639762	115.25409	Jl. By Pass Ngurah Rai
Prof Mantra 1	-8.649863	115.254793	Jl. By Pass Ngurah Rai
Waribang 1	-8.658978	115.253576	Jl. By Pass Ngurah Rai
Matahari Terbit 1	-8.671831	115.258261	Jl. By Pass Ngurah Rai
Sindhu 1	-8.681172	115.259301	Jl. By Pass Ngurah Rai
SMPN 9 1	-8.691389	115.25871	Jl. By Pass Ngurah Rai
Tirtanadi 1	-8.704333	115.25198	Jl. By Pass Ngurah Rai

Danau Poso 1	-8.704337	115.248522	Jl. By Pass Ngurah Rai
Kertapetasikan 1	-8.709496	115.238649	Jl. By Pass Ngurah Rai
Sakenan	-8.713145	115.224629	Jl. By Pass Ngurah Rai
Serangan 1	-8.715136	115.22129	Jl. By Pass Ngurah Rai
Pesanggaran 1	-8.71682	115.213353	Jl. By Pass Ngurah Rai
Pedungan 1	-8.717034	115.205187	Jl. By Pass Ngurah Rai
Pedungan 2	-8.718730	115.200674	Jl. By Pass Ngurah Rai
Tanah Kilap 1	-8.721938	115.189597	Jl. By Pass Ngurah Rai
Dewa Ruci 1	-8.723254	115.181729	Jl. By Pass Ngurah Rai
Dewa Ruci 2	-8.721046	115.18225	Jl. By Pass Ngurah Rai
Raya Kuta	-8.709132	115.181106	Jl. Raya Kuta
Sunset Road Timur	-8.712127	115.186149	Jl. Sunset Road
Patasari 1	-8.738555	115.180093	Jl. By Pass Ngurah Rai
Kelan 1	-8.750494	115.181996	Jl. By Pass Ngurah Rai
Kedonganan 1	-8.762085	115.178834	Jl. By Pass Ngurah Rai
Jimbaran 1	-8.772398	115.17791	Jl. By Pass Ngurah Rai
Jimbaran 2	-8.782314	115.180804	Jl. By Pass Ngurah Rai
Udayana 1	-8.783896	115.188398	Jl. By Pass Ngurah Rai
Mumbul 1	-8.785898	115.203005	Jl. By Pass Ngurah Rai
Bualu 1	-8.794047	115.216269	Jl. By Pass Ngurah Rai
Nusa Dua 1	-8.797896	115.22191	Jl. By Pass Ngurah Rai
BTDC 1	-8.80093425	115.23198076	Jl. Kw Nusa Dua
Nusa Dua – Batu Bulan			
BTDC 1	-8. 80093425	115. 23198076	Jl. Kw Nusa Dua
BTDC 2	-8.79095594	115.22720542	Jl. Kw Nusa Dua
Nusa Dua 2	-8.789508	115.225012	Jl. Pratama Raya
Bualu 2	-8.792725	115.214814	Jl. By Pass Ngurah Rai
Mumbul 2	-8.786114	115.203296	Jl. By Pass Ngurah Rai
Taman Griya 2	-8.784005	115.187433	Jl. By Pass Ngurah Rai
Jimbaran 2	-8.782291	115.180478	Jl. By Pass Ngurah Rai
Jimbaran 1	-8.771571	115.177826	Jl. By Pass Ngurah Rai
Kedonganan 2	-8.760053	115.178947	Jl. By Pass Ngurah Rai
Kelan 2	-8.750045	115.182081	Jl. By Pass Ngurah Rai
Patasari 2	-8.73821	115.179927	Jl. By Pass Ngurah Rai
Dewa Ruci 2	-8.721046	115.18225	Jl. By Pass Ngurah Rai
Raya Kuta	-8.709132	115.181106	Jl. Raya Kuta
Sunset Road Timur	-8.712127	115.186149	Jl. Sunset Road
Tanah Kilap 2	-8.721621	115.18644	Jl. By Pass Ngurah Rai
Pedungan 2	-8.720191	115.19649	Jl. By Pass Ngurah Rai
Pedungan 1	-8.716363	115.206717	Jl. By Pass Ngurah Rai

Pesanggaran 2	-8.716838	115.213833	Jl. By Pass Ngurah Rai
Serangan 2	-8.714541	115.222107	Jl. By Pass Ngurah Rai
Kertapetasikan 2	-8.709496	115.238649	Jl. By Pass Ngurah Rai
Danau Poso 2	-8.704253	115.248514	Jl. By Pass Ngurah Rai
Tirtanadi 2	-8.704256	115.252769	Jl. By Pass Ngurah Rai
SMP 9 2	-8.690826	115.25862	Jl. By Pass Ngurah Rai
Sindhu 2	-8.680479	115.259015	Jl. By Pass Ngurah Rai
Matahari Terbit 2	-8.671789	115.258123	Jl. By Pass Ngurah Rai
Waribang 2	-8.655936	115.253798	Jl. By Pass Ngurah Rai
Prof IB. Mantra 2	-8.650495	115.254561	Jl. By Pass Ngurah Rai
Tohpati 2	-8.640958	115.25447	Jl. By Pass Ngurah Rai
Siulan 2	-8.633764	115.256948	Jl. Wage Rudolf Supratman
Batu Bulan	-8.63156596	115.26092596	Jl. Pudak, Sukawati Kab. Gianyar