PENGEMBANGAN SISTEM REKOMENDASI RUTE PALING OPTIMUM DENGAN ALGORITME *VORONOI CONTINUOUS K NEAREST NEIGHBOR* (VCKNN), *PROGRESSIVE INCREMENTAL NETWORK EXPANSION* (PINE), *VORONOI-BASED NETWORK NEAREST NEIGHBOR* (VN3), BERBASIS *WEBGIS*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Fajar Krisna Wicaksono
NIM: 135150401111051

PROGRAM STUDI SISTEM INFORMASI
JURUSAN SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PENGEMBANGAN SISTEM REKOMENDASI RUTE PALING OPTIMUM DENGAN ALGORITME *VORONOI CONTINUOUS K NEAREST NEIGHBOR (VCKNN), PROGRESSIVE INCREMENTAL NETWORK EXPANSION (PINE), VORONOI-BASED NETWORK NEAREST NEIGHBOR (VN3)*, BERBASIS *WEBGIS*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Fajar Krisna Wicaksono
NIM: 135150401111051

Skripsi ini telah diuji dan dinyatakan lulus pada
31 Juli 2018
Telah diperiksa dan disetujui oleh :

Dosen Pembimbing I	Dosen Pembimbing II
D.Sc.Fatwa Ramdani, S.Si., M.Sc.	Moch. Chandra Saputra, S.Kom, M.T, M.Eng.
NIK. 2016118506191001	NIK. 2016098601061001

Mengetahui
Ketua Jurusan Sistem Informasi

Dr. Eng., Herman Tolle, S.T, M.T.
NIP: 19740823 200012 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 31 Juli 2018

Fajar Krisna Wicaksono

NIM: 135150401111051

KATA PENGANTAR

Dengan menyebut nama Allah SWT Yang Maha Pengasih lagi Maha Penyayang. Segala puji bagi Allah SWT karena atas rahmat, hidayah dan ijinNya-lah penulis dapat menyelesaikan Skripsi yang berjudul "Pengembangan Sistem Rekomendasi Rute Paling Optimum dengan Algoritme Voronoi Continuous K Nearest Neighbor (VCKNN), Progressive Incremental Network Expansion (PINE), Voronoi-Based Network Nearest Neighbor (VN3), Berbasis WebGIS". Shalawat beserta salam semoga senantiasa terlimpah curahkan kepada Nabi Muhammad SAW, kepada keluarganya, para sahabatnya, hingga kepada umatnya hingga akhir zaman, amin. Skripsi ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer Universitas Brawijaya Malang.

Dalam penyusunan dan penulisan skripsi ini tidak terlepas dari bantuan, bimbingan serta dukungan dari berbagai pihak. Oleh karena itu dalam kesempatan ini penulis dengan senang hati menyampaikan terima kasih kepada yang terhormat:

1. Bapak, Ibu, dan seluruh keluarga yang dengan tulus selalu memberi doa, dukungan baik berupa materil maupun non materil, nasihat untuk bagaimana sabar, dan tetap berusaha serta motivasi yang tiada henti untuk memberikan semangat kepada penulis dalam pengerjaan skripsi ini hingga selesai.
2. Bapak Wayan Firdaus Mahmudy, S.Si., M.T., Ph.D. selaku Dekan Fakultas Ilmu Komputer, Universitas Brawijaya.
3. Bapak Dr. Eng. Herman Tolle, S.T., M.T. selaku Ketua Jurusan Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya.
4. Bapak Suprapto, S.T., M.T. selaku Ketua Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Brawijaya.
5. Bapak Fatwa Ramdani, D.Sc, S.Si, M.Sc sebagai dosen pembimbing I dan Bapak Mochamad Chandra Saputra, S.Kom., M.Eng sebagai dosen pembimbing II yang selalu dengan senang hati dan sabar dalam memberikan arahan, masukan, saran dukungan, nasihat, dan motivasi dalam pengerjaan skripsi ini. Serta penulis banyak belajar hal-hal baru selama proses pengerjaan skrispsi kepada beliau.
6. Bapak Aryo Pinandito, S.T., M.MT sebagai dosen penasehat akademik yang selalu memberikan nasehat kepada penulis selama menempuh masa studi.
7. Seluruh Dosen Fakultas Ilmu Komputer yang telah memberikan ilmu kepada penulis dari awal sampai akhir masa studi.
8. Seluruh Civitas Akademika Fakultas Ilmu Komputer Universitas Brawijaya yang telah banyak memberikan dukungan dan bantuan selama masa studi dan penyelesaian skripsi ini.

9. Keluarga Besar Pusat Kajian Geo Informatika yang telah menjadi keluarga dan tempat bertukar informasi bagi penulis untuk menyelesaikan skripsi ini.
10. Serta semua pihak yang telah terlibat dalam pengerjaan skripsi ini yang tidak bisa disebutkan penulis satu persatu.

Semoga Allah SWT memberikan balasan yang berlipat ganda kepada semuanya. Demi perbaikan selanjutnya, saran dan kritik yang membangun akan penulis terima dengan senang hati.

Malang, 31 Juli 2018

Penulis

kw.fajar@gmail.com

ABSTRAK

Dalam kesehariannya, masyarakat di Kota Malang sering melakukan perjalanan dari satu tempat ke tempat lain dengan mempertimbangkan efesiensi waktu perjalanan. Kecamatan Lowokwaru adalah salah satu kecamatan yang cukup padat di Kota Malang, dengan pertumbuhan penduduk yang relatif cepat. Oleh karenanya, sering terjadi kemacetan di titik-titik tertentu dan pada waktu tertentu. Pencarian jalur dapat menjadi salah satu solusi dalam menangani kemacetan pada waktu-waktu tertentu di Kota Malang. Beberapa diatara algoritme untuk menyelesaikan masalah pencarian rute, yakni *Voronoi-Based Continuous k Nearest Neighbor* (VCKNN)*, Progressive Incremental Network Expansion* (PINE)*, Voronoi-Based Network Nearest Neighbor* (VN3)*.* Setiap algoritme memiliki langkah-langkah perhitungan serta model data yang berbeda-beda, hal ini berpengaruh kepada waktu komputasi pada setiap algoritme yang mempengaruhi *response time* dari sistem. Pada penelitian ini menggunakan algoritme VCKNN, PINE dan VN3, pada algoritme tersebut sudah menggunakan poligon *vororoi* untuk melakukan pengelompokan data, sehingga perhitungan jaringan jalan lebih efisien. Pada algoritme VCKNN, PINE dan VN3, proses pencarian rute dibagi menjadi 2 tahap, yang pertama mencari tetangga *voronoi* untuk menyaring data berdasarkan poligon *voronoi*, setelah itu mencari rute pada poligon *voronoi* tersebut. Penelitian ini dilakukan untuk mengetahui berapakah *response time*, jumlah *node* yang dilalui serta hubungan antara *response time* dengan jumlah *node* yang dilalui sehingga diketahui algoritme yang paling efisien dalam masalah pencarian rute. Dari penelitian diketahui bahwa algoritme dengan *response time* tercepat yaitu algoritme VCKNN dengan *response time* 0.071 detik dan algoritme dengan jumlah *node* yang dilalui paling sedikit yaitu algoritme PINE dengan jumlah *node* yang dilalui 17 *node*. Pada perbandingan algoritme VCKNN dengan VN3 didapatkan hasil bahwa jumlah *node* yang dilalui mempengaruhi *response time*, sedangkan pada perbandingan VCKNN dengan PINE dan PINE dengan VN3 *response time* yang dilalui tidak mempengaruhi *response time*, hal ini dikarenakan terdapat proses untuk mencari *move interval* terlebih dahulu pada algoritme PINE. Dari keseluruhan hasil penelitan didapatkan algoritme yang paling efisien dalam penentuan rute optimum adalah algoritme VCKNN.

Kata kunci: Algoritme VCKNN, Algoritme PINE, Algoritme VN3, Poligon *Voronoi*

ABSTRACT

In daily life, people in Malang often travel from one place to another by considering the efficiency of travel time. Lowokwaru is one of the densely populated sub-districts in Malang City, with relatively fast population growth. Therefore, traffic jam often occurs frequently at certain time. Path searching is one of the solutions in handling traffic jam at certain time in Malang. There are several algorithms for solving route searching problems such as Voronoi-Based Continuous k Nearest Neighbor (VCKNN)*, Progressive Incremental Network Expansion* (PINE)*, Voronoi-Based Network Nearest Neighbor* (VN3)*. Every algorithm has different calculation steps and data models, it affects the computing time of every algorithm that affects the response time of the system. This research uses the* VCKNN*,* PINE *and* VN3 *algorithms, it already uses the voronoi polygon to perform data groupings, so the calculation of the road network is more efficient. In the* VCKNN*,* PINE *and* VN3 *algorithms, the route searching process is divided into two stages, the first stage is searching for the voronoi neighbor to filter the data based on the voronoi polygon, after that, it searches the route on the voronoi polygon. This research was conducted to find out the response time, the number of passed node and the relationship between response time and the number of passed node, so the most efficient algorithm in the problem of route searching was known. From this research it was known that the algorithm with the fastest response time was* VCKNN *algorithm with 0.071 seconds response time and the algorithm with the lowest number of passed node was* PINE *algorithm with the number of nodes passed by 17 nodes. In comparison of* VCKNN *and* VN3 *algorithm it was found that the number of passed node affects the response time, whereas in comparison of* VCKNN *with* PINE *and* PINE *with* VN3*, the passed response time did not affect the response time, it happened because there was a process to find the move interval first on the* PINE *algorithm. From the results of the research, the most efficient algorithm in determining the optimum route was the* VCKNN *algorithm.*

Keywords : VCKNN *Algorithm*, PINE *Algorithm*, VN3 *Algorithm*, *Voronoi Polygon*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB 1 PENDAHULUAN

1.1 Latar belakang

Indonesia adalah salah satu dari Negara terpadat di dunia, dengan jumlah penduduk mencapai 237.641.326 jiwa (BPS, 2010). Ada tiga hal yang membuat suatu bangsa menjadi besar dan makmur yakni tanah yang subur, kerja keras dan kelancaran transportasi orang dan barang dari suatu tempat ke tempat lainnya (Alhadar, 2011). Kendaraan menjadi salah satu kebutuhan masyarakat saat ini sebagai media transportasi, di Indonesia sendiri pada tahun 2013 untuk kendaraan jenis sepeda motor mencapai 92.976.240 unit dan untuk mobil berpenumpang 12.599.138 unit (BPS, 2014). Hal ini dikarenakan cepatnya pertambahan kedaraan sebagai media transportasi tidak disertai dengan infrastruktur jalan yang lebih memadai (Wiyono, 2012).

Istilah perkotaan merupakan bentang budaya yang ditimbulkan oleh unsur unsur alami dan non alami dengan gejala pemusatan penduduk yang cukup besar dan corak kehidupan yang bersifat heterogen dan materialistis dibanding dengan daerah di belakangnya, perkotaan terbentuk dengan adanya kegitan ekonomi yang lebih kompleks dibanding daerah sekitarnya (Patriandini, et. al. 2013). Dalam prosesnya, kota menjadi lokasi strategis karena memiliki daya tarik bagi penduduk dari luar daerah, akibatnya arus urbanisasi dari daerah pedesaan ke kota menunjukkan pertumbuhan yang cukup tinggi, sebagai contohnya Kota Malang, pada tahun 2010 tercatat jumlah penduduk sebanyak 820.243 jiwa dan di tahun 2011 sebanyak 894.653 jiwa (Ekawati, 2014).

Jumlah kendaraan bermotor di Kota Malang juga semakin meningkat, hal ini dikarenakan Kota Malang sebagai kota Pendidikan dan Pariwisata, membuat daya tarik tersendiri bagi orang-orang untuk mengadu nasib dan mencari peruntungan dalam memenuhi kebutuhan hidupnya (Arum, 2014). Sebagai jalur utama dari Kota Batu menuju Kota Malang, selain itu terdapat banyaknya universitas-universitas besar di Kota Malang seperti Universitas Brawijaya, Universitas Negeri Malang dan Universitas Islam Negeri Maulana Malik Ibrahim Malang. Kecamatan Lowokwaru adalah salah satu kecamatan yang cukup padat di Kota Malang, dengan pertumbuhan penduduk yang relatif cepat. Oleh karenanya, sering terjadi kemacetan di titik-titik tertentu dan pada waktu tertentu (Rachmawati, 2010).

Beberapa ruas jalan yang kerap dilanda kemacetan panjang dan lama pada Kecamatan Lowokwaru seperti di pertigaan lampu merah Jalan Dinoyo, perempatan lampu merah ITN dan pertigaan jembatan Soekarno Hatta (Aris dan Ashar, 2013). Salah satu penyebabnya adalah menumpuknya kendaraan pada jalan-jalan tertentu, terlebih pada waktu-waktu tertentu menjadi puncak seperti di pagi hari antara jam 06.00-08.00, siang hari antara jam 12.00-14.00 dan sore hari antara jam 16.00-18.00, namun terdapat kesenggangan jalan lain diwaktu yang sama (Patriandini, et. al. 2013). Hal ini

dikarenakan kurangnya informasi tentang jalan-jalan di sekitar Kecamatan Lowokwaru, selain itu masyarakat yang lebih mengutamakan jalan-jalan utama dan rute terdekat, sehingga sering terjadi kemacetan pada suatu titik.

Dalam kesehariannya, masyarakat di Kota Malang sering melakukan perjalanan dari satu tempat ke tempat lain dengan mempertimbangkan efesiensi waktu perjalanan (Budihartono, 2016). Pencarian jalur merupakan salah satu kebutuhan baru masyarakat sehubungan dengan waktu tempuh yang lebih efisien, oleh karena itu pencarian rute menjadi hal yang penting dalam menangani kemacetan pada waktu-waktu tertentu di Kota Malang (Harahap, 2017). Beberapa diatara algoritme untuk menyelesaikan masalah pencarian rute, yakni *Voronoi-Based Continuous k Nearest Neighbor* (VCKNN)*, Progressive Incremental Network Expansion* (PINE)*, Voronoi-Based Network Nearest Neighbor* (VN3)*,* setiap algoritme memiliki langkah-langkah perhitungan serta model data yang berbeda-beda, hal ini berpengaruh kepada waktu komputasi pada setiap algoritme yang mempengaruhi *response time* dari sistem. Pada penelitian sebelumnya algoritme VCKNN, PINE , VN3 hanya menggunakan bobot jarak, sedangkan dari permasalahan yang terjadi di Kota Malang terdapat perbedaan waktu tempuh di beberapa jalan pada waktu yang berbeda-beda.

Algoritme yang digunakan dalam penelitian ini adalah VCKNN, PINE , VN3 karena dalam algoritme tersebut menggunakan *voronoi* poligon dimana poligon tersebut berfungsi untuk membuat perhitungan jaringan jalan menjadi lebih efisien, meskipun data jaringan jalan yang digunakan sangat besar, hal ini dikarenakan perhitungan jalur hanya dilakukan pada setiap segmen jalan di *voronoi* tertentu berdasarkan tetangga dari *voronoi* tersebut (Safar, 2005). Pada data jaringan jalan terdiri dari kumpulan segmen dan *node*, setiap segmen dihubungkan oleh *node*, maka dari itu di setiap persimpangan akan muncul *node*, jumlah dari *node* mempengaruhi waktu kalkulasi yang berhubungan dengan *response time* sistem.

Dari permasalahan diatas, penulis mengembangkan Sistem Rokemendasi Rute Paling Optimum dengan Algoritme VCKNN, PINE , VN3 Berbasis *WebGIS*. Penelitian ini bertujuan untuk menentukan algoritme dengan *response time* terbaik. Dari solusi di atas penulis berharap hasil penelitian ini dapat merekomendasikan algoritme yang paling sesuai dengan permasalahan pencarian rute di Kota Malang khususnya jalan-jalan di Kecamatan Lowokwaru.

1.2 Rumusan masalah

Berdasarkan latar belakang yang telah dijelaskan sebelumnya maka rumusan masalah yang ingin dibahas pada penelitian skripsi ini adalah:

1. Berapakah *response time* dari metode VCKNN, PINE dan VN3?
2. Berapakah *node* yang dilalui dari metode VCKNN, PINE dan VN3?

3. Bagaimana hubungan antara jumlah *node* yang dilalui dengan *response time*?
4. Bagaimana hasil analisis *node* dan *response time* sistem dari setiap metode VCKNN, PINE dan VN3?

1.3 Tujuan

Berdasarkan permasalahan yang dijelaskan sebelumnya maka tujuan dari penelitian skripsi ini adalah :

1. Mengetahui *response time* dalam penentuan rute optimum dari metode VCKNN, PINE dan VN3.
2. Mengetahui *node* yang dilalui dalam penentuan rute optimum dari metode VCKNN, PINE dan VN3.
3. Mengetahui apakah jumlah *node* yang dilalui berpengaruh dengan *response time*.
4. Mengetahui algoritme yang paling efisien dari algoritme VCKNN, PINE dan VN3 dalam penentuan rute optimum.

1.4 Manfaat

Manfaat yang ingin didapat dari penelitian skripsi ini adalah :

1. Mendapatkan algoritme yang paling sesuai dengan permasalahan penelitian ini.
2. Menjadi refrensi tentang kelebihan dan kekurangan algoritme dalam pencarian rute.

1.5 Batasan masalah

Adapun batasan-batasan Masalah dari masalah yang di jelaskan sebelumnya adalah

1. *Study area* hanya mencakup kecamatan Lowokwaru Kota Malang.
2. Penelitian ini mengabaikan penghalang jalan.
3. Seluruh jaringan jalan mengunakan jalan 2 arah.
4. Hasil dari sistem hanya terfokus kepada pemberian rokemendasi rute optimum menggunakan metode VCKNN, PINE, VN3.
5. Metode yang di uji hanya VCKNN, PINE, VN3.
6. Data waktu segmen didapatkan dari *Google Maps API.*
7. Penentuan *generator point* dilakukan secara acak.
8. Variabel yang di jadikan bobot perhitungan adalah variabel waktu tempuh.

1.6 Sistematika pembahasan

Bab I **PENDAHULUAN**

Menjelaskan latar belakang masalah, perumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan.

Bab II **LANDASAN KEPUSTAKAAN**

Menguraikan seluruh teori dasar dan teori penunjang yang berkaitan dengan penelitian ini.

Bab III **METODOLOGI**

Menguraikan tentang langkah-langkah yang dilakukan dalam penelitian.

Bab IV **PERANCANGAN DAN PENGOLAHAN DATA**

Menguraikan tentang perancangan sistem dan pengolahan data jaringan jalan.

Bab V **IMPLEMENTASI**

Menguraikan proses implementasi sesuai dengan perancangan sistem.

Bab VI **PENGUJIAN DAN EVALUASI**

Menguraikan proses pengujian dari sistem serta evaluasi hasil pengujian.

Bab VII **PENUTUP**

Bab ini memuat kesimpulan dan saran.

BAB 2 LANDASAN KEPUSTAKAAN

2.1 Tinjauan Pustaka

Tinjauan pustaka merupakan peninjauan kembali tentang pustaka-pustaka yang terkait dalam penelitian. Tinjauan pustaka dapat berisi tentang penelitian-penelitian terdahulu yang memiliki keterkaitan dengan topik penelitian. Tinjauan pustaka dalam penelitian skripsi ini mengacu kepada penelitian yang dilakukan oleh Zhao, et. al. pada tahun 2011 dengan judul penelitian "*Voronoi-Based Continuous k Nearest Neighbor Search in Mobile Navigation*". Penelitian ini membahas tentang bagaimana menerapkan *algoritme Voronoi-Based Continuous k Nearest Neighbor* untuk pencarian suatu lokasi serta mengolah data-data yang menjadi parameter untuk perhitungan yang dilakukan.

2.2 Sistem Informasi Geografis

Sistem Informasi Geografis (SIG) adalah sistem informasi yang berdasar pada data keruangan dan merepresentasikan obyek di bumi. Dalam SIG sendiri teknologi informasi merupakan perangkat yang membantu dalam menyimpan data, memproses data, menganalisa data, mengelola data dan menyajikan informasi. SIG merupakan sistem yang terkomputerisasi yang menolong dalam mengelola data tentang lingkungan dalam bidang geografis. SIG selalu memiliki relasi dengan disiplin keilmuan Geografi, hal tersebut memiliki hubungan dengan disiplin yang berkenaan dengan yang ada di permukaan bumi, termasuk didalamnya adalah perencanaan dan arsitektur wilayah (Longley, 2001).

Data dalam SIG terdiri atas dua komponen yaitu data spasial yang berhubungan dengan geometri bentuk keruangan dan data attribute yang memberikan informasi tentang bentuk keruangannya (Chang, 2002). Aronoff (1989) menyatakan bahwa SIG adalah sekumpulan komponen yang dilakukan secara manual atau berbasis komputer dimana prosedur-prosedur yang digunakan untuk keperluan menyimpan dan memanipulasi data memiliki referensi geografis. Menurut pendapat tersebut dapat dipahami bahwa, isi aktifitas pada bidang SIG merupakan integrasi dari beragam bidang keilmuan yang didasarkan pada aktifitas SIG tersebut dilakukan.

2.3 *WebGIS*

WebGIS (*Geo Information System*) adalah sistem informasi yang berdasar pada data keruangan dan merepresentasikan obyek di bumi menggunakan teknologi *World Wide Web* yang sering disebut juga sebagai *website* dalam proses perancangan, implementasi dan pengolahan data. Dalam beberapa tahun terakhir, *WebGIS* berkembangan sangat pesat dimana presentasi sederhana sebuah peta yang sebelumnya statis mulai di transformasikan menjadi peta dinamis untuk memudahkan dalam analisis data. Teknologi ini telah dikembangkan oleh instansi perusahaan, pemerintahan, lembaga penelitian dan

masyarakat umum yang digunakan untuk pendukung keputusan, akses data spasial, eksplorasi serta visualisasi data spasial, ruang pengolahan analisis data dan pemodelan serta digunakan untuk mengintegrasikan layanan berbasis geo spasial dengan layanan proses komputasi dan lingkungan (Songnian, 2011).

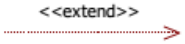
2.4 *Unified Modeling Language* (UML)

UML adalah sebuah bahasa yang menjadi standar dalam merancang dan mendokumentasikan sebuah sistem atau perangkat lunak (Khasanah, 2015). UML menyediakan 9 (sembilan) standar diagram yang dapat digunakan untuk menggambarkan sebuah sistem dari berbagai sudut pandang. Diagram tersebut antara lain : diagram kelas, diagram objek, *use case diagram*, *sequence diagram*, *collaboration diagram*, *state chart diagram*, activity diagram, *component diagram*, *deployment diagram*.

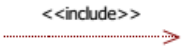
2.4.1 *Use Case Diagram*

Use case diagram didefinisikan sebagai sekumpulan elemen yang menggambarkan fungsionalitas sebuah sistem (Shalahuddin dan Sukamto, 2013). *Use case* merupakan jembatan antara analis dan *user* dalam menentukan gambaran kegunaan dari sebuah sistem. *Use case* diagram memiliki notasi-notasi yang mewakili fungsi-fungsi tertentu seperti yang digambarkan pada Tabel 2.1.

Tabel 2.1 Simbol-Simbol Pada *Use Case Diagram*

Simbol	Nama	Deskripsi
	Aktor	Aktor merepresentasikan peran-peran yang dapat diambil oleh *user* untuk dapat berinteraksi dengan sistem. Aktor tidak selalu berwujud manusia, aktor juga dapat berbentuk sistem lain.
	*Use case*	*Use case* merupakan gambaran dari aksi-aksi yang dapat dilakukan oleh aktor
	Relationship	Relationship menggambarkan hubungan khusus antar elemen yang memiliki makna tertentu
	*Extend Relationship*	*Extend relationship* merupakan relasi yang menggambarkan hubungan sebuah *use case* dengan *use case* tambahan lain yang bersifat *optional*.

Tabel 2.1 Simbol-Simbol Pada *Use Case Diagram* (Lanjutan)

	*Include Relationship*	*Include relationship* merupakan relasi yang mengharuskan sebuah *use case* dijalankan terlebih dahulu agar dapat menjalankan *use case* lain.

Sumber : (Shalahuddin dan Sukamto, 2013)

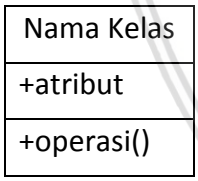
2.4.2 *Use Case Scenario*

Use Case Scenario menggambarkan kegiatan atau peristiwa dari setiap *use case*. Dalam *use case* scenario aktor harus diidentifikasi untuk setiap kegiatan, informasi yang dipertukarkan harus dicatat dan setiap kondisi atau kendala harus tercantum (Presman, 2010).

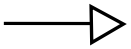
2.4.3 *Class Diagram*

Class Diagram adalah diagram yang menggambarkan struktur sebuah sistem berdasarkan pendefinisian kelas-kelas yang digunakan untuk membangun sistem tersebut (Shalahuddin dan Sukamto, 2013). Kelas memiliki atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang menyusun kelas tersebut, sementara metode adalah fungsi-fungsi yang dapat dijalankan dalam kelas. Tabel 2.2 menjelaskan notasi yang digunakan pada *class diagram*.

Tabel 2.2 Simbol-Simbol Dalam *Class Diagram*

Simbol	Nama	Deskripsi
	Kelas	Himpunan dari objek yang saling berbagi atribut dan metode yang sama.
 nama_interface	*Interface*	Memiliki konsep yang sama dengan *interface* dalam pemrograman berorientasi objek.
	Association	Menggambarkan relasi antar kelas dengan makna yang umum.
	*Directed Association*	Menggambarkan relasi antar kelas dengan makna "kelas yang satu digunakan oleh kelas yang lain".

Tabel 2.2 Simbol-Simbol Dalam *Class Diagram* (Lanjutan)

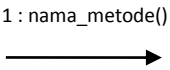
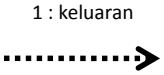
	*Generalization*	Menggambarkan relasi antar kelas dengan makna "generalisasi-spesialisasi (umum-khusus)".
	*Dependency*	Menggambarkan relasi antar kelas dengan makna "kebergantungan sebuah kelas dengan kelas yang lain".

Sumber : (Shalahuddin dan Sukamto, 2013)

2.4.4 *Sequence Diagram*

Sequence diagram adalah diagram yang menggambarkan perilaku objek dengan mendeskripsikan masa hidup objek tersebut serta transaksi *message* antar objek. Sebelum menggambar *sequence diagram*, objek yang terlibat serta metode-metode yang dimiliki kelas yang direpresentasikan menjadi objek tersebut harus diketahui terlebih dahulu (Shalahuddin dan Sukamto, 2013). Tabel 2.3 menjelaskan notasi dari *sequence diagram*.

Tabel 2.3 Simbol-Simbol Dalam *Sequence Diagram*

Simbol	Nama	Deskripsi
	Aktor	Himpunan dari objek yang saling berbagi atribut dan metode yang sama.
	Lifeline	Menyatakan rentan waktu dimana objek dapat hidup di dalamnya.
	Waktu aktif	Menyatakan jika sebuah objek dalam kondisi aktif dan tengah melakukan transaksi.
	*Call*	Menggambarkan jika sebuah objek memanggil operasi atau metode tertentu, baik dari objek lain atau objek itu sendiri.
	*Return*	Menggambarkan sebuah *return* dari suatu objek setelah menjalankan operasi tertentu.

Sumber : (Shalahuddin dan Sukamto, 2013)

2.4.5 *Physical Data Model*

Physical Data Model merupakan presentasi suatu implementasi *database* secara spesifik dari *Logical Data Model* , mencakup detail penyimpanan data di computer yang direpresentasikan dalam bentuk *record format, record ordering dan access path*. *Physical data model* juga menjelaskan bagaimana data itu disimpan dalam media penyimpanan. Tujuannya adalah untuk menciptakan perancangan dalam penyimpanan data yang menyediakan kinerja dan memastikan integritas, keamanan dan kemampuan untuk dipulihkan.

2.5 Topologi

Pada Model GIS, *user* mungkin ingin tahu ketika salah satu daerah yang berkaitan ataupun berada dalam daerah lain, ketika dua daerah berpotongan, atau mungkin hanya ketika dua ruang yang berdekatan satu sama lain. Topologi memberi kita alat untuk menjawab jenis pertanyaan tentang kemungkinan kesalahan yang terjadi dalam mengolah data sehingga dengan demikian memainkan peran sentral dalam pembentukan GIS.

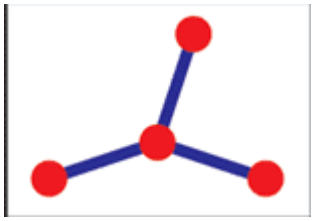
Secara eksplisit, topologi adalah metode untuk mendefinisikan dan membedakan hubungan antara pasangan wilayah geografis ataupun unsur-unsur geografis lainnya. Contohnya Dua negara yang bersebelahan memiliki batasan negara, garis ini disebut *edge*. Topologi sudah menjadi kebutuhan dalam SIG untuk manajemen data dan integritas data. Umumnya, model data topologi mengatur hubungan data spasial dengan merepresentasikan objek data spasial (*point, line* dan *area features*) sebagai sebuah topologi yang lebih tradisional yaitu *nodes, faces* dan *edges*.

Pada dasarnya, topologi digunakan untuk memastikan kualitas data dari model data spasial dan membantu dalam melakukan kompilasi. Topologi juga digunakan untuk menganalisa data spasial dalam beberapa kondisi yang memiliki poligon yangberdekatan dengan nilai atribut yang sama atau berada pada garis yang sama di dalam grafik topologi.

Topologi juga dapat digunakan untuk memodelkan bagaimana geometri dari sejumlah kelas dapat diintegrasikan, fungsi lainnya yakni mendeteksi kesalahan pada peta digital ataupun mendeteksi garis yang tidak terpenuhi dengan benar, poligon yang tidak ditutup dengan benar, dan kesalahan digitalisasi lainnya dalam peta digital. Tiga konsep utama topologi dengan struktur *arc–node* adalah sebagai berikut:

1. *Connectivity* : Segmen garis bersambung satu dengan lainnya dengan perantaraan *node.*
2. *Area definition*: Segmen garis yang saling bersambung yang mengelilingi suatu area / luasan disebut sebagai poligon.
3. *Contiguity* : Segmen garis memiliki arah dan sisi kiri dan sisi kanan.

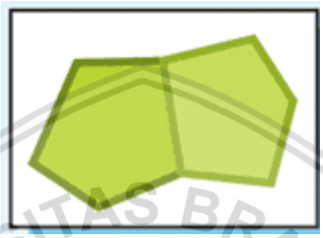
Beberapa jenis topologi sebagai berikut (Esri, 2002) :



Gambar 2.1 *Arc-node topology*

Sumber : (Esri, 2002)

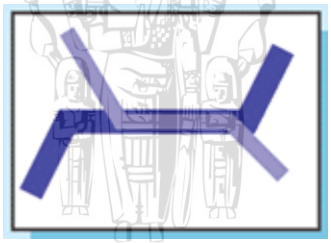
Arc-node topology (Gambar 2.1) adalah objek garis dapat berbagi pada *endpoints* sehingga terhubung dengan objek lainnya.



Gambar 2.2 *Polygon topology*

Sumber : (Esri, 2002)

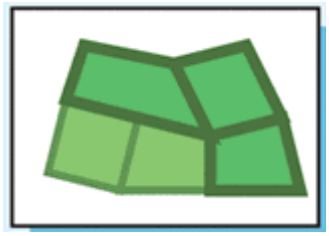
Polygon topology (Gambar 2.2) adalah objek daerah/area dapat berbagi batas, sehingga memisahkan daerah tersebut dengan daerah lainnya.



Gambar 2.3 *Route topology*

Sumber : (Esri, 2002)

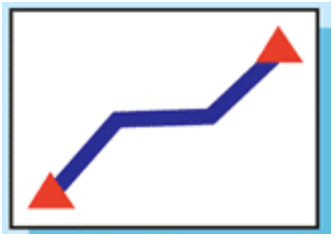
Route topology (Gambar 2.3) adalah objek garis dapat berbagi segmen dengan objek garis lain.



Gambar 2.4 *Region topology*

Sumber : (Esri, 2002)

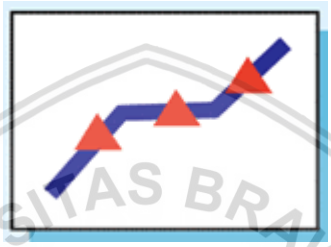
Region topology (Gambar 2.4) adalah objek daerah dapat tumpang tindih dengan objek daerah lainnya.



Gambar 2.5 *Node topology*

Sumber : (Esri, 2002)

Node topology (Gambar 2.5) adalah objek garis dapat berbagi simpul endpoint dengan objek titik



Gambar 2.6 *Point events*

Sumber : (Esri, 2002)

Point events (Gambar 2.6) adalah objek titik dapat berbagi simpul dengan objek fitur.

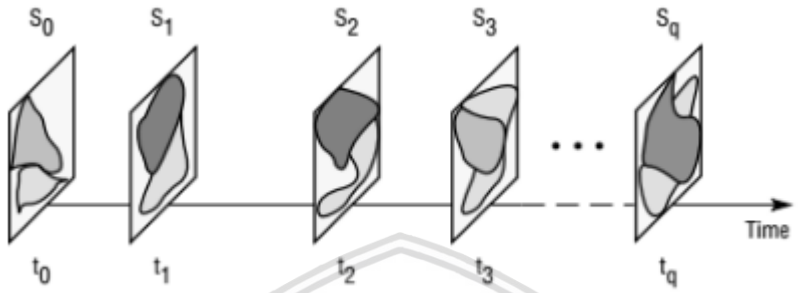
2.6 *Spatio Temporal*

Data diperoleh dengan mengukur beberapa atribut dari suatu entitas/fenomena. ketika atribut ini berkaitan dengan tempat dan waktu, maka data tersebut temasuk *spatio-temporal data*, data *spatio-temporal* dapat digunakan pada berbagai tujuan yang berbeda, pola dalam data *spatio-temporal* mempengaruhi bidang-bidang seperti analisis migrasi, peramalan cuaca, dan *mobile marketing*. Mengelompokan *spatio-temporal data* juga dapat membantu dalam analisis jaringan sosial, yang digunakan dalam tugas-tugas seperti *share data allocation*, iklan dengan target dan lainnya. Dalam langkah ini kita membangun representasi dari data *spatio-temporal* yang sudah melalui *pre-processing*, kemudian mendefinisikan data jumlah berbasis baru berdasarkan kesamaan ukuran antara lintasan untuk menemukan lintasan yang sema dengan kedekatan dalam waktu dan ruang. Oleh karena itu penggukuran *spatio-temporal* data tidak hanya mencakup atribut yang diukur dari nilai-nilai tentang entitas, tetapi juga dua nilai atribut khusus yaitu nilai lokasi dimana pengukuran itu di ambil dan nilai waktu.

2.6.1 *Location Based Representations*

Kemajuan terbaru dalam teknologi komunikasi dan informasi, seperti meningkatnya akurasi teknologi GPS dan mengecilnya perangkat komunikasi nirkabel semakin menunjang layanan *Location-Based Service*. Untuk memperbaiki

layanan tersebut maka digunakan teknik data mining untuk analisis data dalam jumlah yang besar dari data yang di kumpulkan oleh perangkat *mobile*. *Mobile user* diasumsikan perangkat bergerak dimana fitur lokasi dalam keadaan aktif, berarti bahwa aplikasi yang berjalan di perangkat memiliki kemampuan untuk mendapatkan lokasi saat ini, sejarah atau bahkan keadaan masa mendatang.



Gambar 2.7 Pendekatan *Snapshot*

Sumber : (Peuquet, 2017)

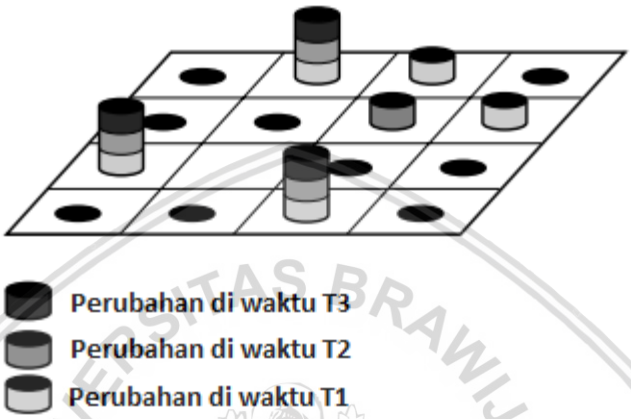
Data dalam GIS harus bisa dilihat sebagai resepresentasi *spatio-temporal*, yaitu serangkaian temporal spasial dalam bentuk *'snapshot'*, seperti yang ditunjukkan pada Gambar 2.7. Pendekatan tersebut untuk mewakili data *spatio-temporal* dimana setiap *'snapshot'* dilambangkan dengan S, dan setiap waktu dilambangkan dengan t.

Organisasi GIS yang menggunakan *database,* data disimpan dalam lapisan tematik. *'snapshot'* adalah pendekatan untuk representasi ruang-waktu biasanya dalam model data *grid*, meskipun model vektor juga dapat digunakan. Dari pada menyimpan semua informasi yang berkaitan dengan domain pembelajaran yang diberikan seperti elevasi atau penggunaan lahan dalam satu lapisan, melainkan setiap lapisan berisi informasi yang berkaitan dengan tematik tunggal domain pada satu waktu tertentu. Data demikian tercatat dalam serangkaian interval. Fitur yang membedakan representasi *snapshot* adalah gambar pada setiap titik waktu tertentu disimpan sebagai gambar lengkap atau *snapshot*. Semuanya termasuk terlepas dari apa yang telah atau belum berubah sejak snapshot sebelumnya, dan jarak temporal antara *snapshot* tidak selalu seragam (Peuquet, 2017).

2.6.2 *Entity Based Representations*

Sebuah modifikasi dari model jaringan yang memungkinkan perubahan individu (yaitu peristiwa atau fenomena) pada waktu dan tempat yang akan dipotret diusulkan dalam konteks SIG pada tahun 1990 dan telah dibuat dalam bentuk prototipe ditahun 1996 (Peuquet, 2017). Model ini juga digunakan dalam analisis desain sirkuit elektronik. Alih-alih merekam hanya nilai tunggal untuk satu piksel daftar *variable-length* (dalam bentuk *digital number*) yang terkait dengan setiap *pixel*. Setiap entri dalam daftar mencatat perubahan di lokasi tertentu dilambangkan dengan nilai baru dan waktu di mana perubahan terjadi. Hal ini

ditunjukkan pada Gambar 2.8 di mana setiap perubahan baru untuk lokasi tertentu ditambahkan ke awal daftar untuk lokasi tersebut. Hasilnya adalah satu set daftar *variable-length* direferensikan ke sel jaringan. Setiap daftar mewakili sejarah peristiwa untuk lokasi sel diurutkan dalam urutan temporal. Contohnya negara dunia untuk seluruh wilayah diambil sebagai nilai pertama lalu disimpan dalam semua daftar lokasional-referensi. Berbeda dengan representasi *snapshot*, representasi ini hanya menyimpan perubahan yang berhubungan dengan lokasi tertentu dan menghindari menyimpan informasi yang berlebihan.



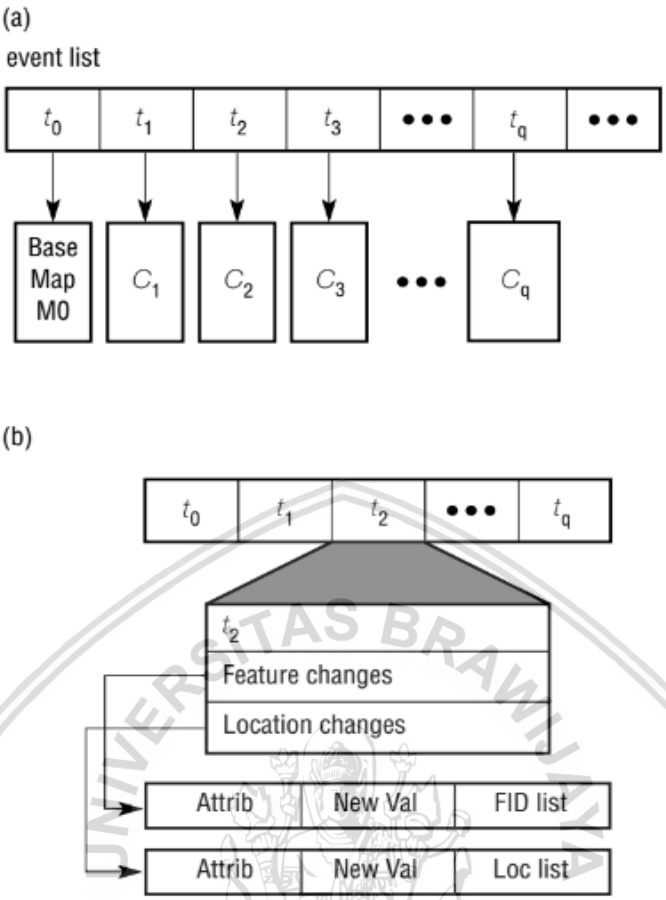
Gambar 2.8 Pendekatan Grid

Sumber : (Peuquet, 2017)

2.6.3 *Time Based Representations*

Semua perubahan disimpan sebagai urutan peristiwa melalui waktu. Waktu yang terkait dengan setiap perubahan disimpan dalam *'world state'* (pada Gambar 2.9). Perbedaan antara kali disimpan menyatakan *interval temporal* antara peristiwa yang berurutan. Perubahan disimpan dalam waktu ini atau 'vektor temporal' dapat berhubungan dengan lokasi, badan, atau keduanya (Gambar 2.9 b). Waktu seperti itu merupakan perkembangan melalui waktu perubahan yang ditentukan dari beberapa tanggal atau peristiwa (t0) ke waktu berikutnya yang ditentukan, hingga titik waktu di kemudian (tn). Setiap lokasi dalam garis waktu (dengan lokasi temporal yang tercatat sebagai t0, t1, ..., tn) dapat berkait dengan set tertentu dari lokasi dan entitas dalam ruang-waktu yang berubah (atau diamati memiliki perubahan) pada waktu tertentu dan notasi perubahan tertentu (Peuquet, 2017).

Dengan jenis representasi berbasis waktu, perubahan yang berkaitan dengan waktu secara eksplisit disimpan. Jenis representasi ini memiliki keuntungan unik yakni memfasilitasi *query* berbasis waktu (misalnya mengambil semua peristiwa yang terjadi antara 1 Januari dan 30 Maret, 1995). Menambahkan peristiwa baru sebagai waktu berjalan juga mudah, perubahan hanya ditambahkan ke akhir timeline. Sebuah garis waktu ordinal memfasilitasi representasi percabangan waktu untuk mewakili urutan alternatif atau paralel peristiwa yang dihasilkan dari kejadian tertentu.



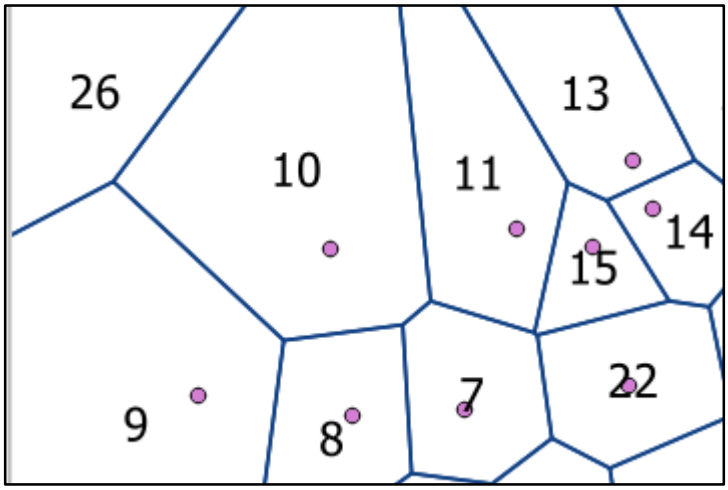
Gambar 2.9 Pendekatan *Time-Line*

Sumber : (Peuquet, 2017)

2.7 *Voronoi Diagram*

Diagram *voronoi* adalah poligon yang terbentuk dari jarak *euclidean* pada satu titik dengan titik lainnya. Diagram *voronoi* dibentuk berdasarkan titik-titik yang disebut dengan *generator point*, setiap titik akan dibatasi dengan poligon dimana setiap titik yang berbatasan memiliki jarak yang sama dengan garis yang membatasi (Zhao, et. al, 2011). Gambar 2.10 merupakan contoh diagram *voronoi*, dimana setiap *generator point* memiliki jarak yang sama.

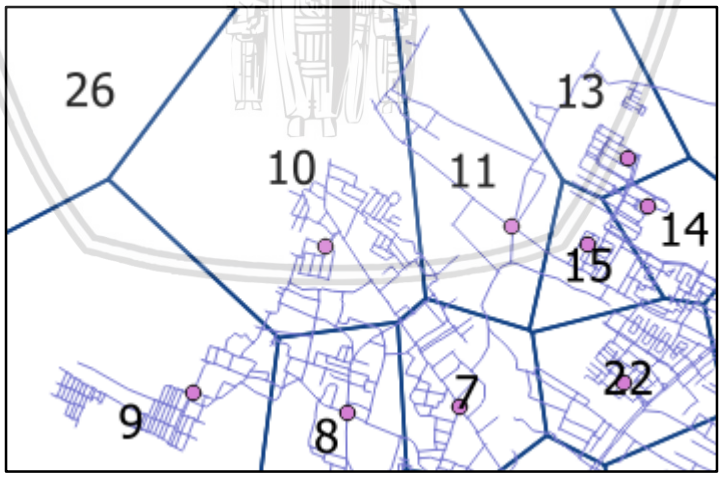
Pada poligon *voronoi*, setiap poligon akan berbatasan dengan poligon lainnya, hal ini disebut dengan tetangga *voronoi*, setiap tetangga memiliki tingkatan masing-masing, semakin banyak *voronoi* yang membatasinya, semakin besar level tetangga. Pada poligon *voronoi* 10 (Gambar 2.10) memiliki tetangga level 1 yaitu poligon 11, poligon 7, poligon 8, poligon 9 dan poligon 26, sedangkan tetangga level 2 yaitu poligon 13, poligon 15 dan poligon 22, dan untuk tetangga level 3 yaiitu poligon 14.



Gambar 2.10 *Voronoi Diagram*

2.7.1 *Diagram Network Voronoi*

Diagram network voronoi adalah jaringan yang dibatasi oleh poligon *voronoi*, hal ini dilakukan untuk mengelompokan jaringan berdasarkan area dari poligon tersebut. Dengan adanya *diagram network voronoi*, data jaringan menjadi memiliki kelompok-kelompok masing-masing, sehingga untuk mengolah data jaringan tidak perlu menggunakan seluruh data, hanya data yang memiliki level tetangga terdekat saja yang digunakan, hal ini dapat membuat data jaringan yang diproses menjadi lebih efisien (Zhao, et. al, 2011). Gambar 2.11 merupakan contoh *diagram network voronoi*, dimana jaringan jalan dikelompokkan berdasarkan poligon *voronoi*.



Gambar 2.11 *Diagram Network Voronoi*

Pada *diagram network voronoi*, setiap poligon akan berbatasan dengan poligon lainnya, namun sedikit berbeda dengan diagram *voronoi*, tetangga *voronoi* hanya *voronoi* yang berbatasan serta memiliki jaringan yang menghubungkan *voronoi* tersebut, setiap tetangga memiliki tingkatan masing-masing, semakin banyak *voronoi* yang membatasinya, semakin besar level tetangga. Pada poligon *voronoi* 10 (Gambar 2.11) memiliki tetangga level 1 yaitu poligon 11, poligon 7,

poligon 8, poligon 9, sedangkan tetangga level 2 yaitu poligon 13, poligon 15 dan poligon 22, dan untuk tetangga level 3 yaiitu poligon 14. Pada poligon 26 bukan tetangga poligon 10 karena tidak ada jaringan yang menghubungkan antara poligon 10 dan poligon 26.

2.8 *Continuous K Nearest Neighbor* (CKNN)

Continuous K nearest neighbor queries (CKNN) didefinisikan sebagai titik terdekat *points of interest* semua titik di jalan (misalnya, mencari tiga SPBU terdekat dengan kendaraaan yang bergerak). Hasil dari *query* adalah sekumpulan interval (atau *split point*) dan KNN objek yang sesuai, sehingga KNNs dari semua objek dalam setiap interval sama (Zhao, et. al, 2011).

2.8.1 *Voronoi Continuous K Nearest Neighbor* (VCKNN)

Voronoi diagram merupakan jenis khusus dari penguraian dari suatu area/daerah yang di tentukan dari jarak ke suatu objek-objek tertentu. Dengan memberikan titik S, lalu akan dihasilkan *diagram voronoi*. Setiap titik memiliki *cell voronoi* sendiri (V), yang terdiri dari semua titik lebih dekat ke S dari pada poin lainya, titik perbatasan antara poligon adalah koleksi poin dengan persamaan dari jarak yang telah dihasilkan.

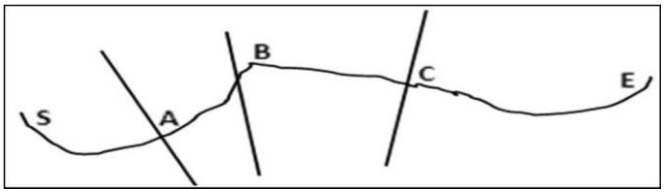
Dalam diagram *voronoi* yang menghitung jarak *euclidean*, perbatasan bersama adalah pertengahan tegak lurus dari link yang terhubung dengan dua generator yang sesuai. Namun, dalam menghitung jarak pada jaringan, batas terdiri dari titik-titik diskrit, yang merupakan titik tengah jaringan jalan yang terhubung dengan dua *generator* yang sesuai. Sebuah poligon dalam jaringan adalah himpunan *node* dan *edges*, yang lebih dekat ke satu *generator* dibandingkan dengan yang lain.

Pada mekanisme *Path-Division*, jumlah segmen ditentukan oleh jumlah poligon *voronoi*. Meskipun ada banyak persimpangan di setiap poligon *voronoi*, metode ini akan memproses setiap poligon *voronoi* sebagai satu unit, sehingga tidak perlu untuk memeriksa setiap persimpangan yang ada. Dalam mencari urutan *Split Nodes* VCKNN menempatkan *split nodes* menggunakan *query-point moving distance*. Untuk setiap interval, VCKNN mengidentifikasi *split node* langsung, yang memiliki jarak terdekat antara titik *query* dan jalur berpotongan/segmen di poligon Voronoi. Dan hasilnya semua *split node* diidentifikasi dalam satu proses. Pada algoritme VCKNN, menggunakan data jaringan jalan pada *voronoi* awal dan *voronoi* akhir, sedangkan pada tetangga *voronoi* menggunakan *move interval* (Zhao, et. al, 2011).

2.8.2 *DAR/EDAR*

DAR / EDAR didasarkan pada teknik *Progressive Incremental Network Expansion* (*PINE*), yang menggunakan jaringan jalan sebagai peta dasar. Pada algoritme ini dimulai dengan membagi jalur/*path* menjadi segmen, yang masing-masing dipisahkan oleh titik persimpangan jaringan. Selanjutnya, mencari tabel KNN untuk dua *node* yang berdekatan, membandingkan dua table, dan menukar posisi untuk membuat keduanya sama. Setiap pertukaran akan membuat *split*

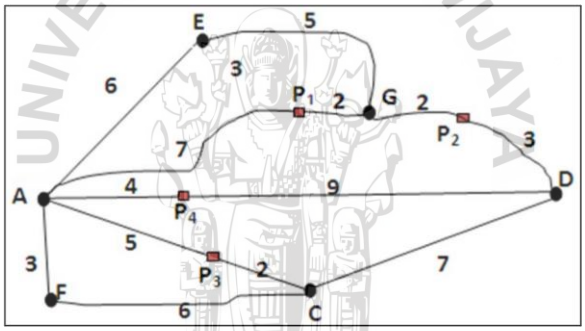
node, dan ketika dua tabel yang persis sama, berarti semua *split node* telah ditemukan. Kemudian, posisi *split node* dan tabel KNN akan menjadi hasil dari *query*. Berikut ini adalah contoh proses *DAR/EDAR*.



Gambar 2.12 langkah pertama dalam DAR

Sumber : (Zhao, et. al, 2011)

Pertama, jalur dibagi menjadi segmen menggunakan *node* yang berpotongan di jalan, contoh *node* yang berpotongan seperti persimpangan jalan, hal ini ditunjukkan pada Gambar 2.12. Dalam contoh ini, *query* dimulai dari S dan berakhir di E. Pada titik S ke titik E memiliki beberapa persimpangan, dan jalan dipisahkan oleh sebuah persimpangan adalah segmen. Dalam contoh ini, pada titik S ke titik A adalah salah satu segmen, dan jalan dari titik A ke titik B adalah segmen, dan sebagainya.



Gambar 2.13 langkah kedua dalam DAR

Sumber : (Zhao, et. al, 2011)

A's list will be

Interest point	Distance
P_4	4
P_3	5

D's list will be

Interest point	Distance
P_2	3
P_1	7

Then create a complete list for A and D

A	Interest point	Distance
PQ	P_4	4
	P_3	5
RQ	P_3	9
	P_3	9

D	Interest point	Distance
PQ	P_2	3
	P_1	7
RQ	P_1	7
	P_2	11

Gambar 2.14 Tabel DAR

Sumber : (Zhao, et. al, 2011)

Kedua, untuk setiap segmen misalnya AD (Gambar 2.13), kita menemukan KNN dari dua titik akhir (A dan D), yang berarti kita menghasilkan dua daftar KNN untuk kedua poin diakhiri (pada Gambar 2.14, menganggap *query* adalah 2NN) dan membentuk satu daftar lengkap (Gambar 2.14). Setelah itu, untuk setiap *interest point* yang berdekatan, untuk menghitung λ seperti persamaan 2.1 (perhatikan bahwa I adalah jarak di RQ untuk titik tujuan tertentu dan Dist adalah fungsi jarak):

$$\lambda_{i,i+1} = \frac{Dist(A,D) + Dist\,(D,I_i') - Dist(A,I_i)}{2} \quad (2.1)$$

Kemudian, menerapkan operasi yang sama antara titik tujuan terakhir dan setiap titik di RQ. λ terkecil akan menjadi arah bergerak dari titik *query*, tukar *list* untuk menemukan *split* lain sampai terdapat dua daftar yang sama(Zhao, et. al, 2011).

Algoritme *Progressive Incremental Network Expansion* (*PINE*) menggunakan *voronoi network diagram* untuk mengelompokan data, setiap segmen dikelompokkan berdasarkan *node* yang muncul pada perbatasan jaringan yang disebut dengan *move interval* (Safar, 2005). Sedikit berbeda dengan algoritme *Voronoi Continuous K nearest neighbor,* Algoritme *Progressive Incremental Network Expansion* menggunakan data *move interval* pada semua *voronoi* dalam pencarian jalur.

2.8.3 *Intersection Examination* (IE)

Intersection Examination (IE) didasarkan pada *Voronoi-based network nearest neighbor* (VN3). Secara umum, mirip dengan EDAR, IE memecah jalur menjadi segmen. IE kemudian mencoba untuk menemukan *split nodes* dengan mendefinisikan jalan untuk setiap titik tujuan dalam daftar hasil KNN saat ini dan dengan mengurutkannya dalam urutan menaik. Ketika ada perubahan dalam posisi titik tujuan, itu menjadi *split node* (Zhao, et. al, 2011).

Pada algoritme *Voronoi-based network nearest neighbor* (VN3) data jaringan di pecah berdasarkan poligon *voronoi*, bebeda dengan algoritme *Voronoi Continuous K nearest neighbor* dan *Progressive Incremental Network Expansion*, pada algoritme *VN3* segmen-segmen dalam poligon *voronoi* tidak dijadikan satu atau tidak dikelompokan, sehingga proses pencarian jalur hanya menggunakan data jaringan tanpa menggunakan data *move interval* (Safar, 2005).

2.9 *Performance Testing*

Performance Testing fokus untuk memastikan tidak ada masalah fungsionalitas pada sistem selama menyediakan kebutuhan non-fungsional dari sistem (Molyneaux, 2015). Salah satu fungsi *Performance Testing* adalah untuk memastikan bahwa sistem sudah memenuhi spesifikasi. Performance Testing dibagi menjadi 2 jenis yakni *service-oriented* dan *efficiency-oriented*. *service-oriented* memiliki indikator *availability* dan *response time* sedangkan *efficiency-oriented* memiliki indikator *throughput* dan *capacity* (Molyneaux, 2015).

Service-oriented berfungsi untuk mengukur seberapa baik (atau tidak) aplikasi menyediakan layanan kepada pengguna. Contoh dari *availability* seperti waktu yang dibutuhkan untuk menampilkan halaman website. Contoh *response time* seperti jumlah waktu dari aplikasi dalam menanggapi permintaan pengguna.

Efficiency-oriented berfungsi untuk mengukur seberapa baik (atau tidak) aplikasi yang menggunakan infrastruktur. Contoh dari *Throughput* yaitu kecepatan sebenarnya pada saat menggunakan aplikasi. Contoh dari *capacity* yaitu memori yang di gunakan disaat 1000 pengguna sedang menggunakan aplikasi.

2.10 Pengujian *Chi-square*

Pengujian *Chi-square* adalah pengujian yang dilakukan untuk melihat apakah variabel yang bersifat *independent* mempengaruhi variabel *dependent* dimana data yang digunakan adalah data nominal (Raharja, 2017). Uji *Chi-square* memiliki persyaratan, yaitu tidak ada kolom dengan nilai 0 pada frekuensi yang diamati, apabila bentuk dari tabel kontingensi adalah 2x2 maka tidak terdapat nilai di bawah 5 pada frekuensi yang diamati, apabila bentuk dari tabel lebih dari 2x2, seperti tabel dengan bentuk 2x3 maka 80% dari nilai frekuensi yang diamati bernilai lebih dari 5 (Hidayat, 2012).

Tahapan awal yang harus dilalukan untuk melakukan uji *Chi-square* adalah membuat tabel kontingensi, Gambar 2.15 merupan contoh tabel kontingensi dengan bentuk 2x3, yang terdiri dari kolom a,b,c,d,e dan f.

Pendidikan	Pekerjaan		Total
	1	2	
1	a	b	a+b
2	c	d	c+d
3	e	f	e+f
Total	a+c+e	b+d+f	N

Gambar 2.15 Tabel Kontingensi

Sumber : (Hidayat, 2012)

Selanjutnya dilakukan pencarian *Chi-square* hitung berdasarkan data dari tabel kontingensi, *Chi-square* hitung didapatkan dengan persamaan 2.2.

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_t} \quad (2.2)$$

Dimana,

O_i= sebagai data yang diamati.

E_i= sebagai data yang diharapkan.

∑= jumlah untuk semua kategori.

Setelah mendapatkan *Chi-square* hitung, dilanjutkan dengan mencari nilai *Chi-square* tabel, *Chi-square* tabel didapatkan berdasarkan nilai α dan derajat bebas, derajat bebas didapatkan dengan persamaan 2.3.

DF=(r-1)x(c-1) (2.3)

Dimana,

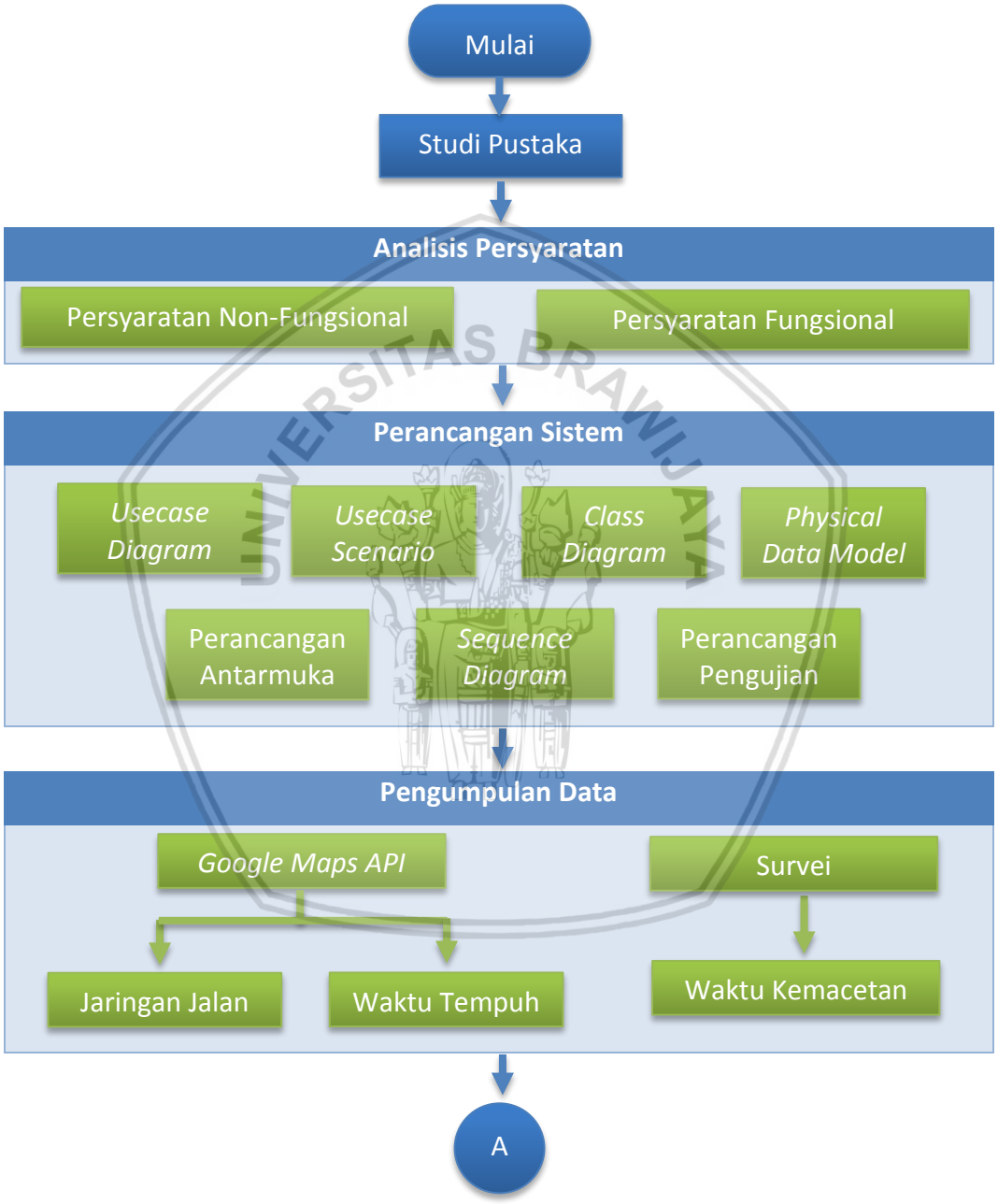
r= jumlah baris pada tabel kontingensi.

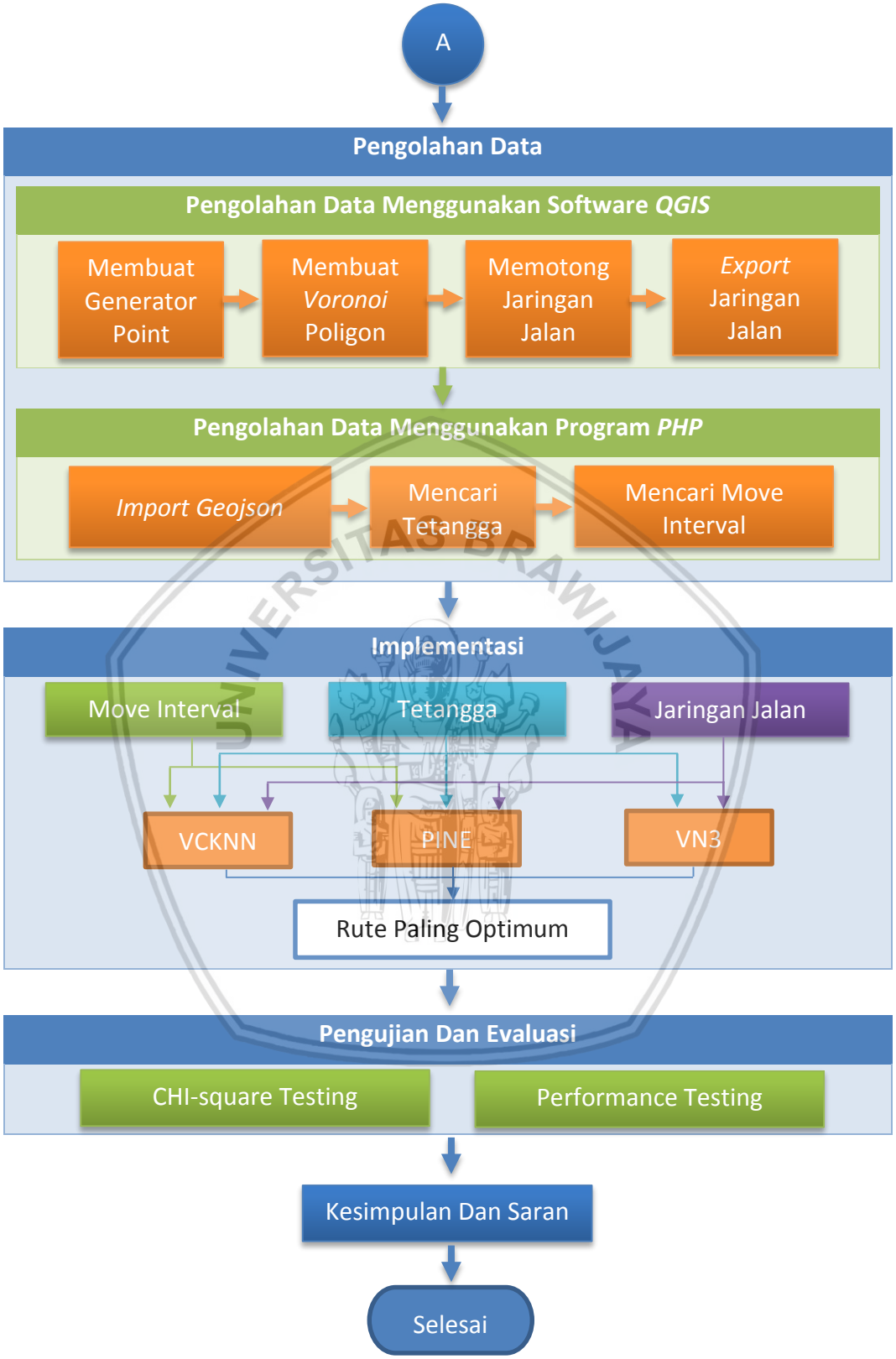
c=jumlah kolom pada tabel kontingensi.

Pengambilan keputusan didapatkan dengan melakukan perbandingan antara nilai *Chi-square* hitung dan *Chi-square* tabel, jika *Chi-square* hitung lebih besar sama dengan dari *Chi-square* tabel, maka hipotesis akhir diterima dan hipotesis awal ditolak.

BAB 3 METODOLOGI

Untuk mencapai tujuan yang telah diuraikan pada Bab 1, maka pada Bab 3 ini peneliti akan menyampaikan metode penelitian yang akan digunakan dalam penelitian ini. Metode penelitian yang digunakan oleh peneliti dapat dilihat pada Gambar 3.1 :





Gambar 3.1 Diagram Alir Metode Penelitian.

3.1 Studi Pustaka

Dalam penelitian ini diawali dengan penulusuran studi pustaka, tujuan penelusuran studi pustaka adalah untuk mendukung serta menunjang dasar teori serta teori-teori pendukung yang digunakan dalam penelitian ini. Penulusuran pustaka dilakukan dengan mempempelajari penelitian sebelumnya, teori mengenai UML, Sistem Informasi Geografis, *WebGIS*, algoritme Dijkstra, algoritme VCKNN, algoritme VN3, algoritme PINE dan pengujian performa.

3.2 Analisis Persyaratan

Tahapan analisis persyaratan bertujuan untuk mendiskripsikan kebutuhan dari sistem yang akan dikembangkan meliputi Persyaratan fungsional dan persyaratan non-fungsional. Berdasarkan latar belakang, masyarakat sering berpergian dari satu tempat ke tempat lain dan menimbulkan permasalahan sering terjadi kemacetan di tempat-tempat dan waktu-waktu tertentu di Kota Malang, sehingga dibutuhkan sebuah sistem pencarian rute paling optimum. Terdapat banyak algoritme yang digunakan untuk melakukan pencarian rute, setiap algoritme memiliki langkah langkah perhitungan dan model data yang berbeda-beda, hal ini berdapak pada *response time* dari sistem pencarian rute.

3.3 Perancangan Sistem

Setelah melakukan anilisis kebutuhan, didapatkan kebutuhan dari sistem yang menjadi acuan dalam perancangan sistem. Perancangan sistem bertujuan untuk menggambarkan sistem yang akan dikembangkan dengan memodelkan menjadi beberapa diagram, yaitu :

1. *Usecase Diagram*
2. *Usecase Scenario*
3. *Sequence Diagram*
4. *Class Diagram*
5. *Physical Data Model*
6. Perancangan Antarmuka
7. Perancangan Pengujian

3.4 Pengumpulan Data

Dalam melaksanakan penelitian ini, dibutuhkan data-data sebagai sumber informasi. Tahapan ini bertujuan untuk mengumpulkan data-data yang digunakan meliputi jaringan jalan Kecamatan Lowokwaru, waktu tempuh setiap segmen, waktu kemacetan dan titik kemacetan.

3.4.1 Jaringan Jalan Kecamatan Lowokwaru

Data jaringan jalan Kecamatan Lowokwaru didapatkan dari digitasi setiap koordinat pada jalan-jalan di Kecamatan Lowokwaru menggunakan *Google Maps*

API. Hasil pada tahapan pendataan jaringan jalan ini adalah jaringan jalan Kecamatan Lowokwaru dalam bentuk *graph* yang di simpan dalam *file shp*. Data jaringan jalan menjadi informasi dasar dalam sistem pencarian rute paling optimum.

3.4.2 Waktu Tempuh Setiap Segmen

Waktu tempuh setiap segmen didapatkan dengan cara melakukan pendataan waktu setiap segmen menggunakan *Google Maps API.* Hasil dari tahapan ini adalah waktu tempuh yang akan dijadikan atribut pada jaringan jalan. Data waktu tempuh digunakan untuk menjadi bobot dalam algoritme VCKNN, VN3 dan PINE.

3.4.3 Waktu Kemacetan

Data waktu serta titik kemacetan didapatkan dengan cara survei kepada pengendara yang sering melewati daerah Lowokwaru (Lampiran B.1) yang di verifikasi oleh pihak Polsek Lowokwaru (Lampiran B.2). Hasil pada tahapan ini adalah titik kemacetan, jam serta hari terjadinya kemacetan. Data waktu kemacetan digunakan untuk melakukan pengelompokan waktu-waktu serta lokasi terjadi kemacetan yang berpengaruh pada waktu tempuh yang lebih lama.

3.5 Pengolahan Data

Pada tahap ini, dilakukan pengolahan data yang telah dikumpulkan dalam tahap pengumpulan data berupa data jaringan jalan beserta atributnya, proses pengolahan di bagi menjadi 2 tahap, yakni pengolahan data menggunakan *software qgis* dan menggunakan program menggunakan bahasa pemrograman php. Tujuan dari tahap ini adalah untuk mencari tetangga dari setiap poligon *voronoi* yang digunakan untuk algoritme VCKNN, VN3 serta PINE dan mencari *move interval* yang digunakan dalam algoritme VCKNN serta PINE. Penjelasan masing masing tahap pada pengolahan data dijelaskan sebagai berikut :

3.5.1 Pengolahan Data Jaringan Jalan menggunakan *Software QGIS*

Berikut ini adalah langkah-langkah dalam pengolahan data menggunakan *software QGIS* :

a. Membuat titik *generator point.*
b. Membuat *voronoi* poligon dari *generator poin.*
c. Memotong jalan berdasarkan *voronoi* poligon.
d. *Export* data jaringan jalan menjadi *WebGIS.*

3.5.2 Pengolahan Data Jaringan Jalan Menggunakan Program Dengan Bahasa Pemrograman *PHP*

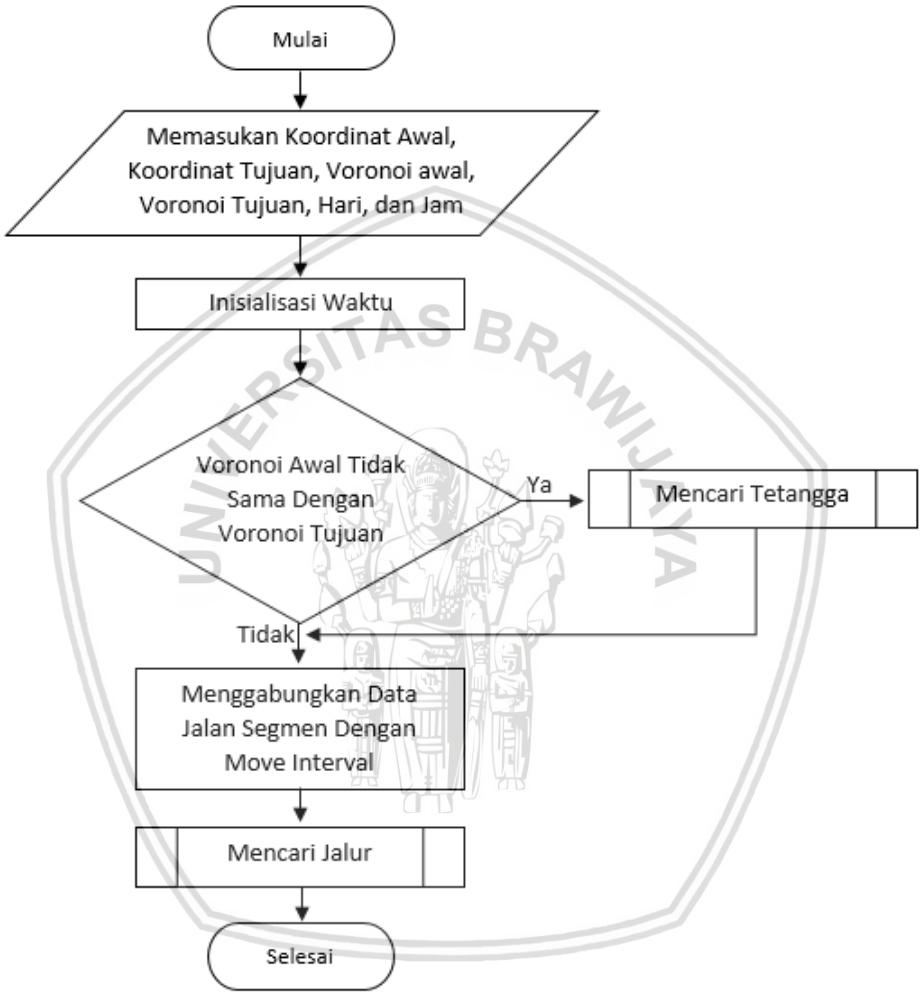
Berikut ini adalah langkah-langkah dalam pengolahan data jaringan jalan menggunakan program dengan bahasa pemrograman *PHP* :

a. *Import* data *geojson* kedalam *database.*
b. Mencari tetangga *voronoi* dan menyimpannya ke *database.*
c. Mencari *move interval* dan menyimpannya ke *database.*

3.6 Implementasi

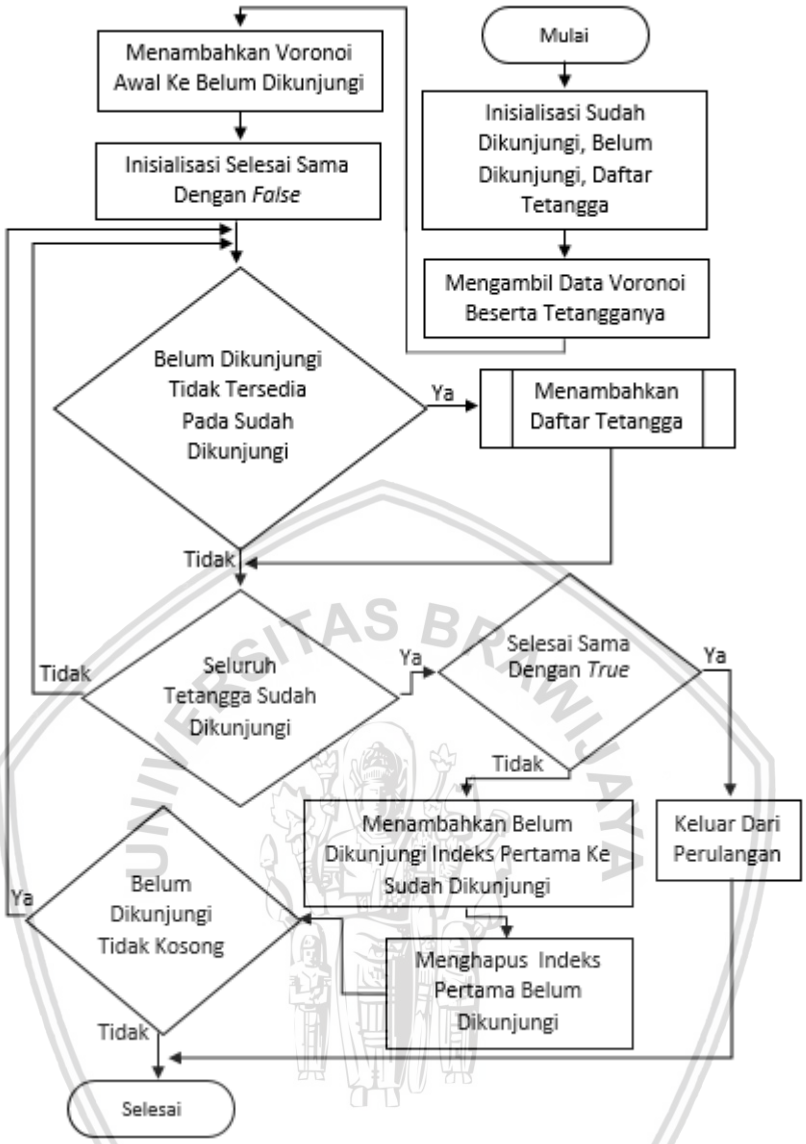
Implementasi sistem informasi yang di buat dilakukan mengacu kepada analisis kebutuhan serta perancangan sistem. Tujuan pada tahapan implementasi adalah untuk mecari rute paling optimum menggunakan algoritme VCKNN, PINE dan VN3. Masing-masing algoritme akan dijelaskan sebagai berikut :

3.6.1 Algoritme VCKNN



Gambar 3.2 *Flowchart* Algoritme VCKNN

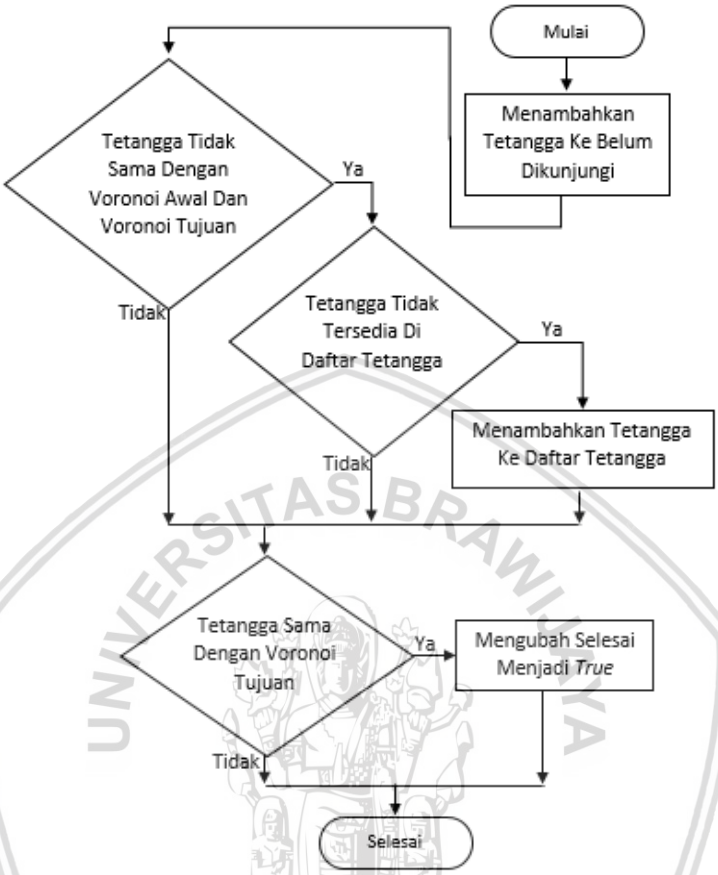
Pada algoritme VCKNN (Gambar 3.2), dimulai dari pengguna memasukan koordinat awal, koordinat tujuan, *voronoi* awal, *voronoi* tujuan, hari dan jam. Dari data hari dan jam, akan dicari terlebih dahulu waktu, setelah itu dilakukan pengecekan *voronoi*, jika *voronoi* berbeda maka akan mencari tetangga *voronoi* terlebih dahulu (Gambar 3.3 dan Gambar 3.4), lalu data *move interval* berdasarkan tetangga *voronoi* di gabung dengan data segmen pada *voronoi* awal dan tujuan, setelah data digabungkan, proses selanjutnya adalah mencari jalur (Gambar 3.5) dan hasil dari algoritme VCKNN adalah rute dengan bobot waktu terkecil.



Gambar 3.3 Subproses Mencari Tetangga.

Pada subproses mencari tetangga (Gambar 3.3), diawali dengan mengambil data *voronoi* beserta tetangganya dari *database* dan menyimpan dalam bentuk variabel *array* multidimensi. Setelah itu menambahkan *voronoi* awal ke belum di kunjungi dan inisialisasi variabel selesai dengan nilai *false*, variabel selesai berfungsi untuk mencari tetangga *local voronoi*. Selanjutnya dilakukan pengecekan apakah belum dikunjungi tidak tersedia pada sudah dikunjungi, jika belum tersedia maka daftar tetangga akan di tambahkan, lalu dilakukan pengecekan apakah seluruh tetangga sudah dikunjungi, jika ada yang belum dikunjungi maka kembali melakukan pengecekan apakah belum dikunjungi tidak tersedia pada sudah dikunjungi, jika seluruh tetangga sudah dikunjungi dilakukan pengecekan apakah selesai bernilai *true*, jika selesai bernilai *true* maka keluar dari perulangan, jika tidak bernilai *true* maka akan menambahkan belum dikunjungi di indeks pertama ke sudah dikunjungi dan menghapus indeks pertama belum dikunjungi. Setelah itu dilakukan pengecekan apakah belum dikunjungi tidak

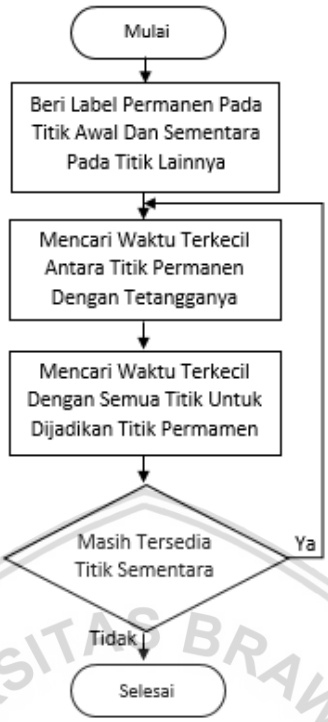
kosong, jika tidak kosong maka kembali melakukan pengecekan belum dikunjungi tidak tersedia pada sudah di kunjungi, jika belum dikunjungi sudah kosong maka keluar dari subproses mencari tetangga.



Gambar 3.4 Subproses Menambah Daftar Tetangga.

Pada subproses menambah daftar tetangga (Gambar 3.4), dimulai dengan menambahkan tetangga ke belum dikunjungi, lalu melakukan pengecekan apakah tetangga tidak sama dengan *voronoi* awal dan *voronoi* tujuan, jika tidak sama akan melakukan pengecekan apakah tetangga tidak tersedia di daftar tetangga, jika tidak tersedia tetangga ditambahkan ke daftar tetangga. Setelah itu di lakukan pengecekan tetangga sama dengan *voronoi* tujuan, jika tetangga sama dengan *voronoi* tujuan maka mengubah nilai selesai menjadi *true* dan subproses menambahkan daftar tetangga selesai.

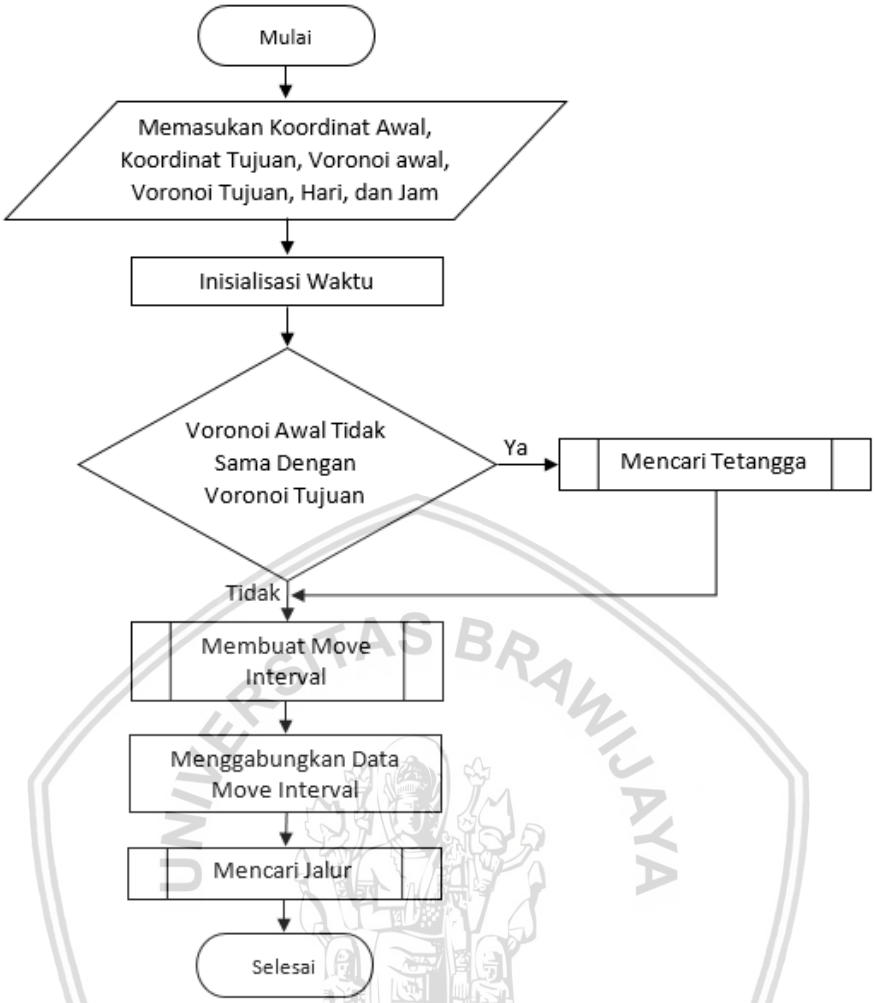
Pada subproses mencari jalur (Gambar 3.5), dimulai dari memberi label permanen pada titik awal dan sementara pada titik lainnya, lalu mencari waktu terkecil antara titik permamen dengan tetangganya dan mencari waktu terkecil dengan semua titik untuk dijadikan titik permanen. Setelah itu melakukan pengecekan apakah masih tersedia titik sementara, jika masih tersedia titik sementara maka kembali mencari waktu terkecil antara titik permanen dengan tetangganya, jika tidak tersedia titik sementara makan subproses mencari jalur selesai.



Gambar 3.5 Subproses Mencari Jalur.

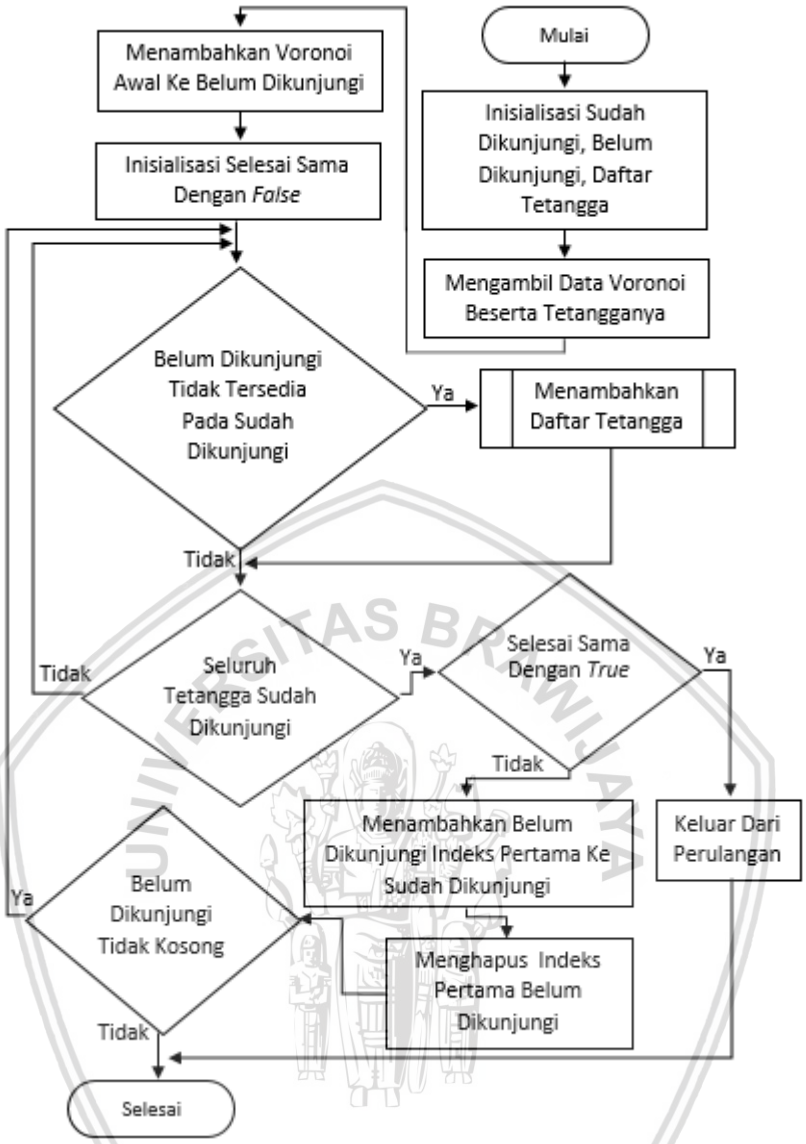
3.6.2 Algoritme PINE

Pada algoritme PINE (Gambar 3.6), dimulai dari pengguna memasukan koordinat awal, koordinat tujuan, *voronoi* awal, *voronoi* tujuan, hari dan jam. Dari data hari dan jam, akan dicari terlebih dahulu waktu, setelah itu dilakukan pengecekan *voronoi*, jika *voronoi* berbeda maka akan mencari tetangga *voronoi* terlebih dahulu (Gambar 3.7 dan Gambar 3.8). Setelah itu membuat *move interval* (Gambar 3.9) pada *voronoi* awal (Gambar 3.11) dan *voronoi* tujuan (Gambar 3.10). Selanjutnya data *move interval* berdasarkan tetangga *voronoi* di gabung dengan data *move interval* pada *voronoi* awal dan tujuan, setelah data digabungkan, proses selanjutnya adalah mencari jalur (Gambar 3.12) dan hasil dari algoritme PINE adalah rute dengan bobot waktu terkecil.



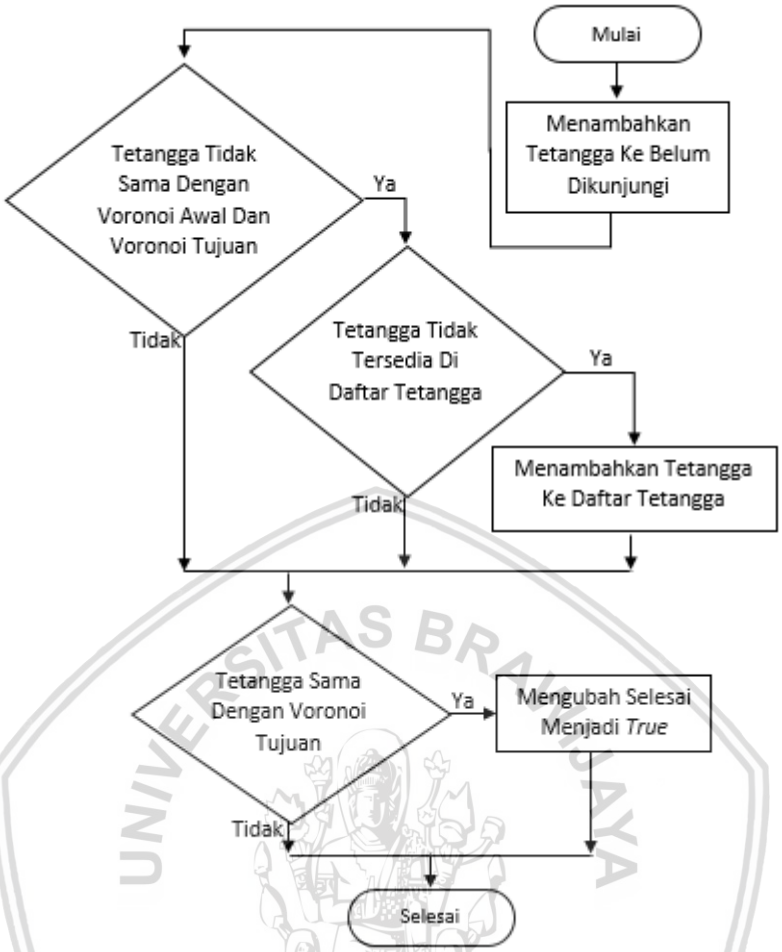
Gambar 3.6 *Flowchart* Algoritme PINE

Pada subproses mencari tetangga (Gambar 3.7), diawali dengan mengambil data *voronoi* beserta tetangganya dari *database* dan menyimpan dalam bentuk variabel *array* multidimensi. Setelah itu menambahkan *voronoi* awal ke belum di kunjungi dan inisialisasi variabel selesai dengan nilai *false*, variabel selesai berfungsi untuk mencari tetangga *local voronoi*. Selanjutnya dilakukan pengecekan apakah belum dikunjungi tidak tersedia pada sudah dikunjungi, jika belum tersedia maka daftar tetangga akan di tambahkan, lalu dilakukan pengecekan apakah seluruh tetangga sudah dikunjungi, jika ada yang belum dikunjungi maka kembali melakukan pengecekan apakah belum dikunjungi tidak tersedia pada sudah dikunjungi, jika seluruh tetangga sudah dikunjungi dilakukan pengecekan apakah selesai bernilai *true*, jika selesai bernilai *true* maka keluar dari perulangan, jika tidak bernilai *true* maka akan menambahkan belum dikunjungi di indeks pertama ke sudah dikunjungi dan menghapus indeks pertama belum dikunjungin. Setelah itu dilakukan pengecekan apakah belum dikunjungi tidak kosong, jika tidak kosong maka kembali melakukan pengecekan belum dikunjungi tidak tersedia pada sudah di kunjungi, jika belum dikunjungi sudah kosong maka keluar dari subproses mencari tetangga.



Gambar 3.7 Subproses Mencari Tetangga.

Pada subproses menambah daftar tetangga (Gambar 3.8), dimulai dengan menambahkan tetangga ke belum dikunjungi, lalu melakukan pengecekan apakah tetangga tidak sama dengan *voronoi* awal dan *voronoi* tujuan, jika tidak sama akan melakukan pengecekan apakah tetangga tidak tersedia di daftar tetangga, jika tidak tersedia tetangga ditambahkan ke daftar tetangga. Setelah itu di lakukan pengecekan tetangga sama dengan *voronoi* tujuan, jika tetangga sama dengan *voronoi* tujuan maka mengubah nilai selesai menjadi *true* dan subproses menambahkan daftar tetangga selesai.

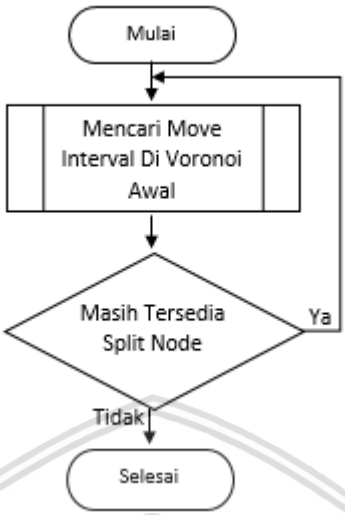


Gambar 3.8 Subproses Menambah Daftar Tetangga.

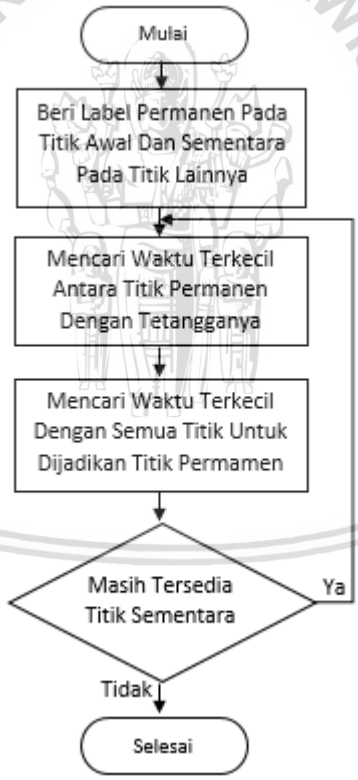
Gambar 3.9 Subproses Membuat Move Interval.

Pada subproses membuat *move interval* (Gambar 3.9), dimulai dengan mencari *split node* pada *voronoi* awal dan *voronoi* tujuan, lalu mencari *move interval* di

voronoi awal dan mencari *move interval* di *voronoi* tujuan, setelah itu subproses membuat *move interval* selesai.



Gambar 3.10 Subproses Mencari Move Interval Di Voronoi Tujuan.



Gambar 3.11 Subproses Mencari Jalur.

Pada subproses mencari *move interval* di *voronoi* tujuan (Gambar 3.10) dimulai dengan mencari *move interval* di *voronoi* awal, lalu melakukan pengecekan apakah masih tersedia *split node*, jika *split node* masih tersedia kembali mencari *move interval* di *voronoi* awal, jika *split node* sudah tidak tersedia subproses mencari *move interval* di *voronoi* tujuan selesai.

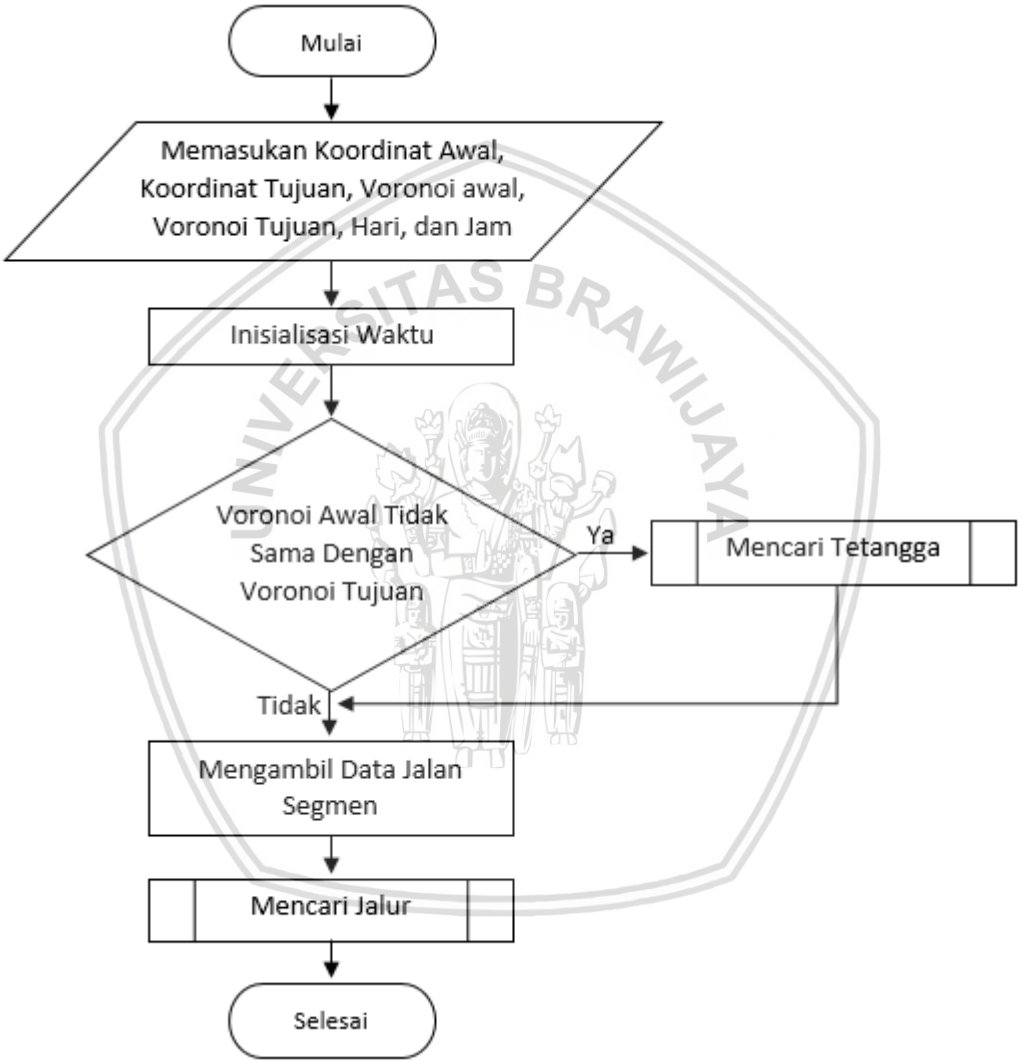
Pada subproses mencari *move interval* di *voronoi* awal (Gambar 3.12), dimulai dengan inisialisasi waktu, nilai waktu di dapatkan dari hari dan jam, lalu mengambil data jalan segen dari *database*, selanjutnya memberi label permanen pada titik awal dan sementara pada titik lainnya, lalu mencari waktu terkecil antara titik permamen dengan tetangganya dan mencari waktu terkecil dengan semua titik untuk dijadikan titik permanen. Setelah itu melakukan pengecekan apakah masih tersedia titik sementara, jika masih tersedia titik sementara maka kembali mencari waktu terkecil antara titik permanen dengan tetangganya, jika tidak tersedia titik sementara makan subproses mencari *move interval* di *voronoi* awal selesai.

Gambar 3.12 Subproses Mencari Move Interval Di Voronoi Awal.

Pada subproses mencari jalur (Gambar 3.11), dimulai dari memberi label permanen pada titik awal dan sementara pada titik lainnya, lalu mencari waktu terkecil antara titik permamen dengan tetangganya dan mencari waktu terkecil dengan semua titik untuk dijadikan titik permanen. Setelah itu melakukan pengecekan apakah masih tersedia titik sementara, jika masih tersedia titik sementara maka kembali mencari waktu terkecil antara titik permanen dengan tetangganya, jika tidak tersedia titik sementara makan subproses mencari jalur selesai.

3.6.3 Algoritme VN3

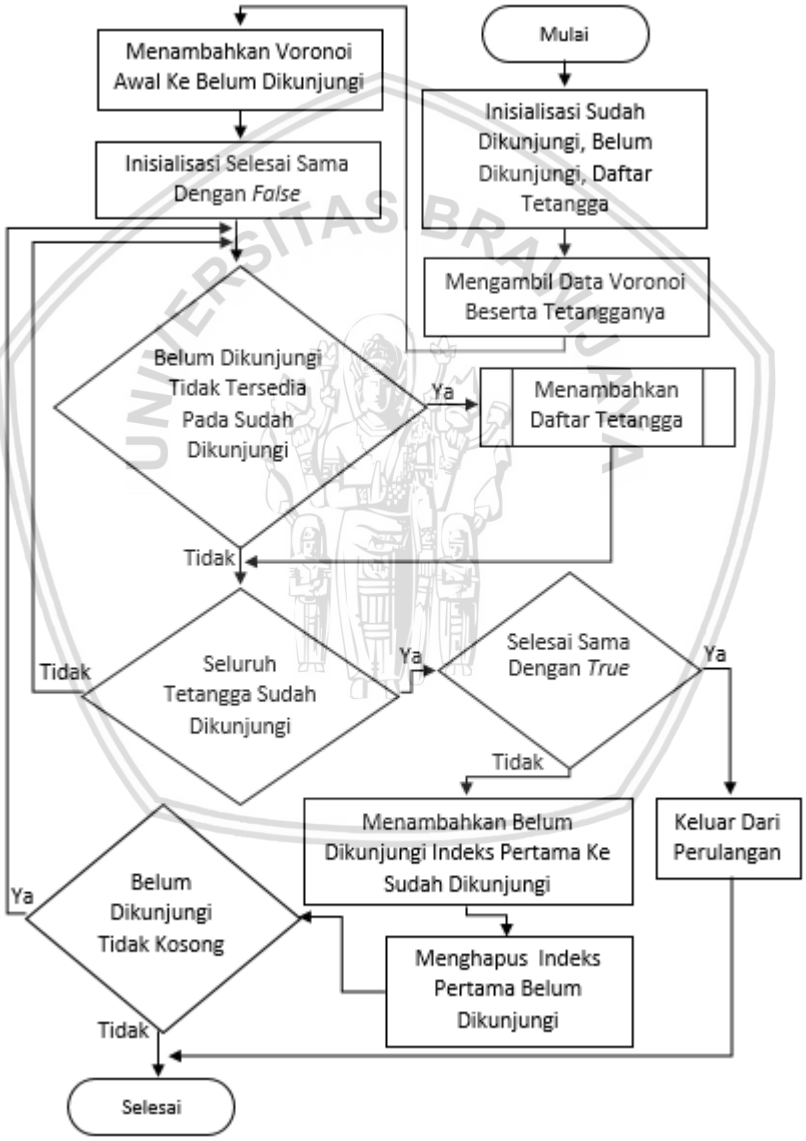
Pada algoritme VN3 (Gambar 3.13), dimulai dari pengguna memasukan koordinat awal, koordinat tujuan, *voronoi* awal, *voronoi* tujuan, hari dan jam. Dari data hari dan jam, akan dicari terlebih dahulu waktu, setelah itu dilakukan pengecekan *voronoi*, jika *voronoi* berbeda maka akan mencari tetangga *voronoi* terlebih dahulu (Gambar 3.14 dan Gambar 3.15), setelah itu mengambil data jalan segmen dan proses pencarian jalur (Gambar 3.16), hasil dari algoritme VN3 adalah rute dengan bobot waktu terkecil.



Gambar 3.13 *Flowchart* Algoritme VN3

Pada subproses mencari tetangga (Gambar 3.14), diawali dengan mengambil data *voronoi* beserta tetangganya dari *database* dan menyimpan dalam bentuk variabel *array multidimensi*. Setelah itu menambahkan *voronoi* awal ke belum di kunjungi dan inisialisasi variabel selesai dengan nilai *false*, variabel selesai berfungsi untuk mencari tetangga *local* voronoi. Selanjutnya dilakukan pengecekan apakah belum dikunjungi tidak tersedia pada sudah dikunjungi, jika belum tersedia maka daftar tetangga akan di tambahkan, lalu dilakukan

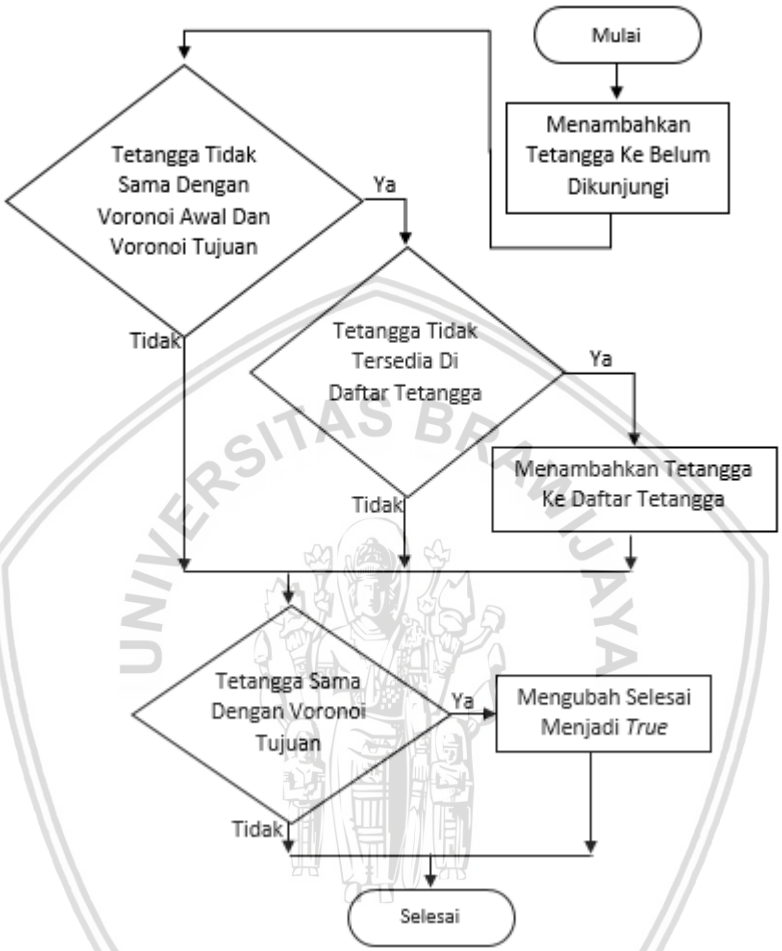
pengecekan apakah seluruh tetangga sudah dikunjungi, jika ada yang belum dikunjungi maka kembali melakukan pengecekan apakah belum dikunjungi tidak tersedia pada sudah dikunjungi, jika seluruh tetangga sudah dikunjungi dilakukan pengecekan apakah selesai bernilai *true*, jika selesai bernilai *true* maka keluar dari perulangan, jika tidak bernilai *true* maka akan menambahkan belum dikunjungi di indeks pertama ke sudah dikunjungi dan menghapus indeks pertama belum dikunjungi. Setelah itu dilakukan pengecekan apakah belum dikunjungi tidak kosong, jika tidak kosong maka kembali melakukan pengecekan belum dikunjungi tidak tersedia pada sudah di kunjungi, jika belum dikunjungi sudah kosong maka keluar dari subproses mencari tetangga.



Gambar 3.14 Subproses Mencari Tetangga.

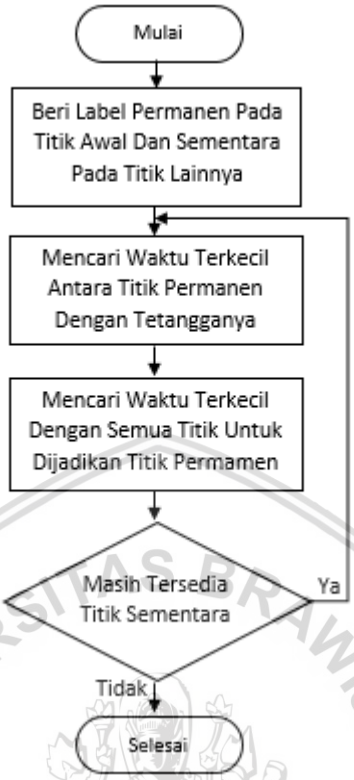
Pada subproses menambah daftar tetangga (Gambar 3.15), dimulai dengan menambahkan tetangga ke belum dikunjungi, lalu melakukan pengecekan apakah tetangga tidak sama dengan *voronoi* awal dan *voronoi* tujuan, jika tidak sama akan

melakukan pengecekan apakah tetangga tidak tersedia di daftar tetangga, jika tidak tersedia tetangga ditambahkan ke daftar tetangga. Setelah itu di lakukan pengecekan tetangga sama dengan *voronoi* tujuan, jika tetangga sama dengan *voronoi* tujuan maka mengubah nilai selesai menjadi true dan subproses menambahkan daftar tetangga selesai.



Gambar 3.15 Subproses Menambah Daftar Tetangga.

Pada subproses mencari jalur (Gambar 3.16), dimulai dari memberi label permanen pada titik awal dan sementara pada titik lainnya, lalu mencari waktu terkecil antara titik permamen dengan tetangganya dan mencari waktu terkecil dengan semua titik untuk dijadikan titik permanen. Setelah itu melakukan pengecekan apakah masih tersedia titik sementara, jika masih tersedia titik sementara maka kembali mencari waktu terkecil antara titik permanen dengan tetangganya, jika tidak tersedia titik sementara makan subproses mencari jalur selesai.



Gambar 3.16 Subproses Mencari Jalur.

3.7 Pengujian dan Evaluasi

Pengujian sistem pencarian rute paling optimum dilakukan agar dapat menunjukan bahwa sistem telah mampu bekerja sesuai dengan spesifikasi dari kebutuhan sistem yang melandasinya. Pengujian dilakukan dengan *performance testing* pada aspek *response time* sistem, pegujian performa dilakukan untuk mencari berapakah *response time* dari algoritme VCKNN, PINE dan VN3, dan mengetahui *node* yang dilalui pada masing masing masing algoritme.

Response time dan jumlah *node* yang dilalui dijadikan data untuk evaluasi. Evaluasi pada sistem pencarian rute paling optimum menggunakan uji *Chi-square*, dimana dengan algoritme dengan jumlah *node* yang dilalui lebih sedikit dijadikan hipotesis awal. Evaluasi dilakukan untuk mencari hubungan antara *response time* dan jumlah *node* yang dilalui serta melakukan analisis *response time* dan jumlah *node* yang dilalui pada algoritme VCKNN, PINE dan VN3.

3.8 Kesimpulan dan Saran

Tahapan terakhir dalam penelitian ini ada pengambilan kesimpulan dan saran. Kesimpulan menjelaskan jawaban dari pertanyaan penelitian. Setelah mendapatkan kesimpulan dilanjutkan dengan memberikan saran untuk pengembangan lebih lanjut dari sistem yang di kembangkan.

BAB 4 PERANCANGAN SISTEM DAN PENGOLAHAN DATA

Pada perancangan sistem pencarian rute paling optimum akan menjelaskan tentang daftar terstruktur kebutuhan perangkat lunak secara fungsional dan non-fungsional. Pada bab persyaratan dan kebutuhan juga dijelaskan mengenai perancangan *usecase* yang menggambarkan hubungan antara pengguna dengan sistem dan apa saja yang dapat dilakukan pengguna terhadap sistem. Kemudian pada perancangan akan dibuat sebuah perancangan UML yang terdiri dari perancangan *Secuence Diagram* untuk menggambarkan alur kerja dari sistem, *Class Diagram* untuk mengetahui struktur dari kelas-kelas yang terdapat dalam sistem.

4.1 Analisis Persyaratan

4.1.1 Identifikasi Tipe Pemangku Kepentingan

Tahapan identifikasi tipe pemangku kepentingan bertujuan untuk mendeskripsikan serta mengelompokkan pemangku kepentingan sesuai dengan karakteristik dari pemangku kepentingan dan hubungan antara sistem dengan pemangku kepentingan. Tabel 4.1 menjelaskan tipe pemangku kepentingan dalam sistem pencarian rute paling optimum.

Tabel 4.1 Tipe Pemangku Kepentingan

Tipe Pemangku Kepentingan	Deskripsi	Pemangku Kepentingan
Pengguna	Individu yang berinteraksi secara langsung dengan sistem yang berperan sebagai aktor pada *use case*.	Pengguna.
Pengembang	Organisasi atau individu yang mengembangkan sistem.	Peneliti.

4.1.2 Analisis Masalah

Setelah mengelompokkan pemangku kepentingan, dilanjutkan dengan melakukan analisis masalah yang bertujuan untuk memahami masalah yang terjadi serta solusi yang harus dilakukan. Analisis masalah dalam penelitian ini diidentifikasi berdasarkan masalah yang muncul dari latar belakang. Masyarakat sering berpergian dari satu tempat ke tempat lain dan menimbulkan permasalahan sering terjadi kemacetan di tempat-tempat dan waktu-waktu tertentu di Kecamatan Lowokwaru, sehingga dibutuhkan sebuah sistem pencarian rute paling optimum. Setiap algoritme yang digunakan untuk pencarian rute memiliki langkah-langkah perhitungan serta model data yang berbeda-beda, hal ini berdampak pada *response time* dari sistem. Dari masalah yang sudah dijelaskan dilakukan pengembangan sistem pencarian rute paling optimum menggunakan

algoritme VCKNN, PINE dan VN3 untuk mengetahui algoritme yang paling sesuai dengan keadaan di Kecamatan Lowokwaru.

4.1.3 Identifikasi Kebutuhan Pengguna

Setelah menemukan solusi dari permasalahan yang muncul, dilanjutkan dengan identifikasi kebutuhan pengguna. Kebutuhan pengguna menjelaskan pernyataan-pernyataan yang memiliki hubungan dengan masalah yang sudah diidentifikasi pada tahapan analisis masalah. Hasil pada tahapan identifikasi kebutuhan pengguna digunakan sebagai informasi dasar untuk analisis fitur. Tabel 4.2 menjelaskan tentang identifikasi kebutuhan pengguna dalam sistem pencarian rute paling optimum. Setiap kebutuhan pengguna diberikan kode sebagai identitas, keterangan kodefikasi kebutuhan pengguna sebagai berikut.

Kode Kebutuhan Pengguna: SPR-N-xx

Keterangan :
SPR : Singkatan dari Sistem Pencarian Rute
N : Kode representasi kebutuhan
XX : Nomor urut kebutuhan

Tabel 4.2 Hasil Identifikasi Kebutuhan Pengguna

Kode Kebutuhan	Kebutuhan Pengguna	Pemangku Kepentingan	Deskripsi
SPR-N-1	Sistem dapat menampilkan peta untuk algoritme VCKNN	Pengguna	Pengguna dapat melihat peta Kecamatan Lowokwaru.
SPR-N-2	Sistem dapat menampilkan peta untuk algoritme PINE	Pengguna	Pengguna dapat melihat peta Kecamatan Lowokwaru.
SPR-N-3	Sistem dapat menampilkan peta untuk algoritme VN3	Pengguna	Pengguna dapat melihat peta Kecamatan Lowokwaru.
SPR-N-4	Sistem dapat Mencarikan Rute menggunakan algoritme VCKNN	Pengguna	Pengguna dapat melihat rekomendasi rute paling optimum dari algoritme VCKNN.
SPR-N-5	Sistem dapat Mencarikan Rute menggunakan algoritme PINE	Pengguna	Pengguna dapat melihat rekomendasi rute paling optimum dari algoritme PINE.

Tabel 4.2 Hasil Identifikasi Kebutuhan Pengguna (Lanjutan)

SPR-N-6	Sistem dapat Mencarikan Rute menggunakan algoritme VN3	Pengguna	Pengguna dapat melihat rekomendasi rute paling optimum dari algoritme VN3.

4.1.4 Identifikasi Pengguna

Tahapan identifikasi pengguna bertujuan untuk mengetahui individu yang akan menggunakan sistem secara langsung. Hasil dari identifikasi pengguna diperoleh berdasarkan kategori pengguna dalam pemangku kepentingan yang akan menjadi aktor pada diagram *use case*. Tabel 4.3 menjelaskan tentang identifikasi pengguna dalam sistem pencarian rute paling optimum.

Tabel 4.3 Hasil Identifikasi Pengguna

Tipe Pemangku Kepentingan	Tipe Pengguna	Deskripsi
Pengguna	Pengguna	Individu yang menggunakan sistem pencarian rute paling optimum pada waktu-waktu yang ditentukan.

4.1.5 Identifikasi Fitur

Identifikasi fitur merupakan tahap untuk menjelaskan solusi yang ditawarkan dalam memenuhi kebutuhan pengguna sehingga dapat menyelesaikan masalah yang muncul. Hasil identifikasi fitur menjelaskan layanan-layanan yang harus disediakan oleh sistem untuk memenuhi kebutuhan pengguna. Setiap fitur diberikan kode sebagai identitas, keterangan kodefikasi fitur sebagai berikut.

Kode Kebutuhan Pengguna: FEAT xx

Keterangan : FEAT : Kode representasi fitur

XX : Nomor urut fitur

Tabel 4.4 Hasil Identifikasi Fitur

Kode Fitur	Deskripsi
FEAT 1	Sistem dapat menampilkan peta serta jaringan jalan pada Kecamatan Lowokwaru.
FEAT 2	Sistem dapat merekomendasikan jalur paling optimum pada waktu waktu tertentu.

Hasil dari tahapan ini digunakan untuk identifikasi persyaratan fungsional serta persyaratan non-fungsional dari sistem pencarian rute paling optimum. Informasi mengenai identitas dan deskripsi fitur dijelaskan pada Tabel 4.4.

Setiap fitur berhubungan dengan kebutuhan pengguna, Tabel 4.5 menjelaskan pengelompokkan kebutuhan pengguna berdasarkan keterkaitan kebutuhan pengguna dengan fitur. Pengelompokkan ini mendefinisikan bahwa sebuah fitur menjadi solusi dari kebutuhan pengguna.

Tabel 4.5 Hubungan Kebutuhan Pengguna dengan Fitur

Kebutuhan Pengguna	Fitur
SPR-N-1	FEAT 1
SPR-N-2	
SPR-N-3	
SPR-N-4	FEAT 2
SPR-N-5	
SPR-N-6	

4.1.6 Persyaratan Fungsional

Pada tahapan ini setiap persyaratan fungsional sistem akan dijelaskan pada Tabel 4.6. Persyaratan fungsional sistem menjelaskan kemampuan-kemampuan yang disediakan oleh sistem sehingga dapat menjalankan fitur-fitur pada sistem serta memenuhi kebutuhan pengguna. Setiap persyaratan fungsional diberikan kode sebagai identitas, keterangan kodefikasi persyaratan fungsional sebagai berikut.

Kode Kebutuhan Pengguna: SPR-PF-xx

Keterangan :
SPR : Singkatan dari Sistem Pencarian Rute
PF : Kode representasi persyaratan fungsional
XX : Nomor urut persyaratan fungsional

Tabel 4.6 Persyaratan Fungsional

Kode Fitur	Kode Persyaratan Fungsional	Deskripsi
FEAT 1	SPR-PF-01	Pengguna dapat melihat peta Kecamatan Lowokwaru.
	SPR-PF-02	Pengguna dapat melihat jaringan jalan Kecamatan Lowokwaru.
FEAT 2	SPR-PF-03	Pengguna dapat mendefinisikan waktu saat ini secara otomatis.
	SPR-PF-04	Pengguna dapat mendefinisikan waktu secara manual.

Tabel 4.6 Persyaratan Fungsional (Lanjutan)

	SPR-PF-05	Pengguna dapat memilih lokasi awal.
	SPR-PF-06	Pengguna dapat memilih lokasi tujuan.
	SPR-PF-07	Sistem dapat merekomendasikan rute paling optimum.

4.1.7 Persyaratan Non-Fungsional

Persyaratan Non-Fungsional adalah pengelompokan persyaratan-persyaratan sistem yang tidak ada kaitannya dari fungsional sistem. Pada Tabel 4.7 merupakan kebutuhan non-fungsional dari sistem pencarian rute paling optimum. Setiap persyaratan non-fungsional diberikan kode sebagai identitas, keterangan kodefikasi persyaratan non-fungsional sebagai berikut.

Kode Kebutuhan Pengguna: SPR-NF-xx

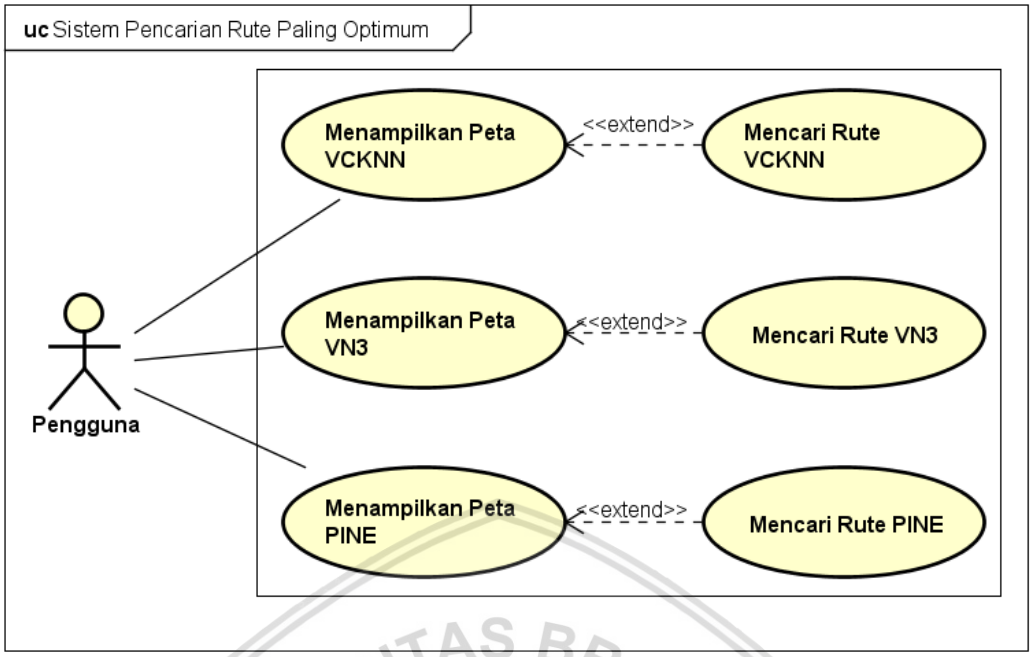
Keterangan :
SPR : Singkatan dari Sistem Pencarian Rute
NF : Kode representasi persyaratan non-fungsional
XX : Nomor urut persyaratan non-fungsional

Tabel 4.7 Persyaratan Non-Fungsional

Kode Persyaratan Non-Fungsional	Nama Kebutuhan	Deskripsi
SPR-NF-01	*Response time*	Waktu yang dibutuhkan sistem untuk menanggapi permintaan pengguna.

4.2 Pemodelan *Use Case Diagram*

Setelah melakukan analisis persyaratan, tahapan selanjutnya adalah melakukan pemodelan *use case diagram.* Diagram *use case* adalah diagram yang digunakan untuk mengetahui perilaku sistem. Diagram *usecase* terdiri dari sekumpulan *usecase*, aktor dan hubungannya. *Usecase* merupakan fungsionalitas dari sistem yang diinisiasi oleh aktor. Setiap pengguna dikelompokkan berdasarkan tujuan pengguna menggunakan sistem. Gambar 4.1 merupakan gambaran diagram *use case* dari sistem pencarian rute paling optimum :



Gambar 4.1 Diagram *Use Case*

Dalam sistem pencarian rute paling optimum, teridentifikasi 6 *use case* dan 1 aktor. Kemudian dilakukan penghubungan antara *use case* dengan pemangku kepentingan, hal ini dilakukan supaya pemangku kepentingan dapat memahami fungsi-fungsi dari sistem pencarian rute paling optimum. Tabel 4.8 menjelaskan hubungan pemangku kepentingan dengan *use case* sitem pencarian rute paling optimum.

Tabel 4.8 Hubungan *Use Case* dengan Pemangku Kepentingan

Use Case	**Pengguna**	**Tipe Pemangku Kepentingan**
Menampilkan Peta VCKNN	Pengguna	Pengguna
Menampilkan Peta VN3		
Menampilkan Peta PINE		
Mencari Rute VCKN		
Mencari Rute VN3		
Mencari Rute PINE		

Selain berhubungan dengan pemangku kepentingan, *use case* berhubungan juga dengan fitur-fitur sistem pencarian rute paling optimum. Hal ini menunjukkan bahwa hasil analisis persyaratan digunakan untuk dasar pemodelan *use case*. Tabel 4.9 menjelaskan hubungan *use case* dengan fitur sistem pencarian rute paling optimum.

Tabel 4.9 Hubungan *Use Case* dengan Fitur

Use Case	Fitur
Menampilkan Peta VCKNN	FEAT 1
Menampilkan Peta VN3	
Menampilkan Peta PINE	
Mencari Rute VCKNN	FEAT 2
Mencari Rute VN3	
Mencari Rute PINE	

4.2.1 Deskripsi Aktor

Pada tahapan deskripsi aktor, akan dijelaskan siapa saja pengguna sistem serta hal-hal yang dapat dilakukannya. Tabel 4.10 menjelaskan aktor yang terdapat dalam sistem pencarian rute paling optimum.

Tabel 4.10 Deskripsi Aktor

Nama Aktor	Deskripsi
Pengguna	Aktor Pengguna diperankan oleh seseorang yang menggunakan Sistem. Aktor pengguna dapat Melihat Peta VCKNN, Melihat Peta VN3, Melihat Peta PINE, Mencari Rute VCKNN, Mencari Rute VN3 dan Mencari Rute PINE.

4.2.2 Use Case Scenario

Use case scenario adalah penjelasan lebih rinci dari setiap *use case*. Pada sistem pencarian rute paling optimum memiliki 6 *use case*, pada tahapan ini masing masing dari *use case* akan dijelaskan mengenai nomor kebutuhan dari setiap *use case*, deskripsi *use case*, aktor yang menjalankannya, kondisi saat *use case* belum dijalankan, hal yang dilakukan pada *use case*, alternatif lain jika terdapat kondisi pada *use case* dan kondisi saat *use case* sudah dijalankan. Berikut ini merupakan skenario dari masing-masing *use case* yang telah dijelaskan pada gambar 4.1.

4.2.2.1 *Use Case Scenario* Menampilkan Peta VCKNN

Pada *use case* Menampilkan Peta VCKNN dengan nomor kebutuhan SPR-N-1, bertujuan untuk memberikan informasi jaringan jalan pada Kecamatan Lowokwaru, *use case* ini juga memberikan informasi peta pada Kecamatan Lowokwaru dan sekitarnya untuk mempermudah pengguna dalam memilih lokasi awal dan lokasi tujuan. Tabel 4.11 menjelaskan skenario dari *use case* Menampilkan Peta VCKNN.

Tabel 4.11 *Use Case Scenario* Menampilkan Peta VCKNN

Nomor Kebutuhan	SPR-N-1
Deskripsi	*Use case* Menampilkan Peta VCKNN bertujuan untuk memberikan informasi peta serta jaringan jalan di Kecamatan Lowokwaru.
Aktor	Pengguna.
Pre-Condition	• Pengguna terhubung dengan internet. • Pengguna terhubung dengan sistem pencarian rute paling optimum.
Main-Flow	1. Pengguna memasukan *url* sistem. 2. Sistem menampilkan peta.
Alternative-Flow	-
Post-Condition	• Pengguna dapat melihat jaringan jalan serta peta Kecamatan Lowokwaru.

4.2.2.2 *Use Case Scenario* Menampilkan Peta VN3

Pada *use case* Menampilkan Peta VN3 dengan nomor kebutuhan SPR-N-2, bertujuan untuk memberikan informasi jaringan jalan pada Kecamatan Lowokwaru, *use case* ini juga memberikan informasi peta pada Kecamatan Lowokwaru dan sekitarnya untuk mempermudah pengguna dalam memilih lokasi awal dan lokasi tujuan. Tabel 4.12 menjelaskan skenario dari *use case* Menampilkan Peta VN3.

Tabel 4.12 *Use Case Scenario* Menampilkan Peta VN3

Nomor Kebutuhan	SPR-N-2
Deskripsi	*Use case* Menampilkan Peta VN3 bertujuan untuk memberikan informasi peta serta jaringan jalan di Kecamatan Lowokwaru.
Aktor	Pengguna.
Pre-Condition	• Pengguna terhubung dengan internet. • Pengguna terhubung dengan sistem pencarian rute paling optimum.
Main-Flow	1. Pengguna memasukan *url* sistem. 2. Sistem menampilkan peta.
Alternative-Flow	-

Tabel 4.12 *Use Case Scenario* Menampilkan Peta VN3 (Lanjutan)

Post-Condition	• Pengguna dapat melihat jaringan jalan serta peta Kecamatan Lowokwaru.

4.2.2.3 *Use Case Scenario* Menampilkan Peta PINE

Pada *use case* Menampilkan Peta PINE dengan nomor kebutuhan SPR-N-3, bertujuan untuk memberikan informasi jaringan jalan pada Kecamatan Lowokwaru, *use case* ini juga memberikan informasi peta pada Kecamatan Lowokwaru dan sekitarnya untuk mempermudah pengguna dalam memilih lokasi awal dan lokasi tujuan. Tabel 4.13 menjelaskan skenario dari *use case* Menampilkan Peta PINE.

Tabel 4.13 *Use Case Scenario* Menampilkan Peta PINE

Nomor Kebutuhan	SPR-N-3
Deskripsi	*Use case* Menampilkan Peta PINE bertujuan untuk memberikan informasi peta serta jaringan jalan di Kecamatan Lowokwaru.
Aktor	Pengguna.
Pre-Condition	• Pengguna terhubung dengan internet. • Pengguna terhubung dengan sistem pencarian rute paling optimum.
Main-Flow	1. Pengguna memasukan *url* sistem. 2. Sistem menampilkan peta.
Alternative-Flow	-
Post-Condition	• Pengguna dapat melihat jaringan jalan serta peta Kecamatan Lowokwaru.

4.2.2.4 *Use Case Scenario* Mencari Rute VCKNN

Pada *use case* Mencari Rute VCKNN dengan nomor kebutuhan SPR-N-4, bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme VCKNN berdasarkan waktu, lokasi awal dan lokasi tujuan yang ditentukan oleh pengguna. Tabel 4.14 menjelaskan skenario dari *use case* Mencari Rute VCKNN.

Tabel 4.14 *Use Case Scenario* Mencari Rute VCKNN

Nomor Kebutuhan	SPR-N-4
Deskripsi	*Use case* Mencari Rute VCKNN bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme VCKNN berdasarkan waktu, lokasi awal dan lokasi tujuan yang ditentukan oleh pengguna.
Aktor	Pengguna
Pre-Condition	• Pengguna terhubung dengan internet. • Sistem Menampilkan Peta VCKNN.
Main-Flow	1. Pengguna mendefinisikan waktu • Waktu Otomatis : ***Varian 1a*** • Waktu Manual : ***Varian 1b*** 2. Pengguna memasukkan lokasi awal. 3. Pengguna memasukkan lokasi tujuan. 4. Pengguna menekan tombol cari jalur. 5. Sistem menampilkan rute.
Alternative-Flow	***Variant 1a****:* Waktu Otomatis 1. Pengguna memilih waktu saat ini. ***Variant 1b:*** Waktu Manual 1. Pengguna memilih waktu manual. 2. Pengguna memasukkan hari. 3. Pengguna memasukkan jam.
Post-Condition	• Pengguna dapat melihat rute paling optimum yang di rekomendasikan menggunakan algoritme VCKNN.

4.2.2.5 *Use Case Scenario* Mencari Rute VN3

Pada *use case* Mencari Rute VN3 dengan nomor kebutuhan SPR-N-5, bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme VN3 berdasarkan waktu, lokasi awal dan lokasi tujuan yang ditentukan oleh pengguna. Tabel 4.15 menjelaskan skenario dari *use case* Mencari Rute VN3.

Tabel 4.15 *Use Case Scenario* Mencari Rute VN3

Nomor Kebutuhan	SPR-N-5
Deskripsi	*Use case* Mencari Rute VN3 bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme VN3 berdasarkan waktu, lokasi awal dan lokasi tujuan yang ditentukan oleh pengguna.
Aktor	Pengguna
Pre-Condition	• Pengguna terhubung dengan internet. • Sistem Menampilkan Peta VN3.
Main-Flow	1. Pengguna mendefinisikan waktu • Waktu Otomatis : ***Varian 1a*** • Waktu Manual : ***Varian 1b*** 2. Pengguna memasukkan lokasi awal. 3. Pengguna memasukkan lokasi tujuan. 4. Pengguna menekan tombol cari jalur. 5. Sistem menampilkan rute.
Alternative-Flow	***Variant 1a:*** Waktu Otomatis 1. Pengguna memilih waktu saat ini. ***Variant 1b:*** Waktu Manual 1. Pengguna memilih waktu manual. 2. Pengguna memasukkan hari. 3. Pengguna memasukkan jam.
Post-Condition	• Pengguna dapat melihat rute paling optimum yang di rekomendasikan menggunakan algoritme VN3.

4.2.2.6 *Use Case Scenario* Mencari Rute PINE

Pada *use case* Mencari Rute PINE dengan nomor kebutuhan SPR-N-6, bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme PINE berdasarkan waktu, lokasi awal dan lokasi tujuan yang ditentukan oleh pengguna. Tabel 4.16 menjelaskan skenario dari *use case* Mencari Rute PINE.

Tabel 4.16 *Use Case Scenario* Mencari Rute PINE

Nomor Kebutuhan	SPR-N-6
Deskripsi	Use case Mencari Rute PINE bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme PINE berdasarkan waktu, lokasi awal dan lokasi tujuan yang ditentukan oleh pengguna.
Aktor	Pengguna
Pre-Condition	• Pengguna terhubung dengan internet. • Sistem Menampilkan Peta PINE.
Main-Flow	1. Pengguna mendefinisikan waktu • Waktu Otomatis : ***Varian 1a*** • Waktu Manual : ***Varian 1b*** 2. Pengguna memasukkan lokasi awal. 3. Pengguna memasukkan lokasi tujuan. 4. Pengguna menekan tombol cari jalur. 5. Sistem menampilkan rute.
Alternative-Flow	***Variant 1a:*** Waktu Otomatis 1. Pengguna memilih waktu saat ini. ***Variant 1b:*** Waktu Manual 1. Pengguna memilih waktu manual. 2. Pengguna memasukkan hari. 3. Pengguna memasukkan jam.
Post-Condition	• Pengguna dapat melihat rute paling optimum yang di rekomendasikan menggunakan algoritme PINE.

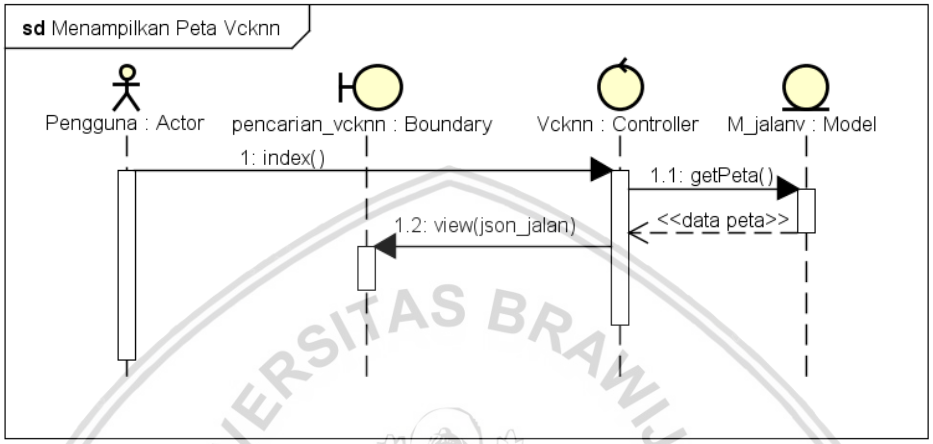
4.3 Perancangan *Sequence Diagram*

Pada tahapan ini dilakukan perancangan *sequence diagram* untuk sistem pencarian rute paling optimum, perancangan *sequence diagram* digunakan untuk memvisualisasikan interaksi antar objek serta pertukaran pesan antara objek *controller*, objek model dan entitas *boundary*. Pembuatan *sequence diagram* didasarkan dari alur pada *use case scenario* dan kelas-kelas dari perancangan kelas diagram.

4.3.1 *Sequence Diagram* Menampilkan Peta VCKNN

Gambar 4.2 merupakan *sequence diagram* Menampilkan Peta VCKNN dengan nomor kebutuhan SPR-N-1 yang bertujuan untuk memberikan informasi peta

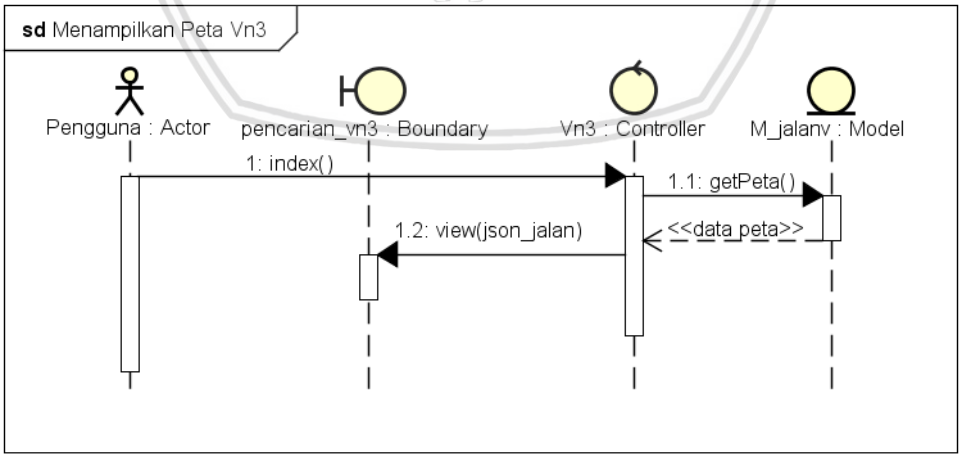
serta jaringan jalan di Kecamatan Lowokwaru. *Sequence diagram* Menampilkan Peta *VCKKN* melibatkan aktor pengguna, *boundary* pencarian_vcknn, *controller* Vcknn dan *model* M_jalanv. *Sequence diagram* Menampilkan Peta *VCKKN* dimulai saat pengguna memasukkan *url* sistem pencarian rute paling optimum dimana *url* tersebut akan meneruskan permintaan kepada kelas Vcknn, setelah itu dilanjutkan dengan mengambil data jaringan jalan menggunakan fungsi getPeta pada *model* M_jalanv, lalu menampilkan informasi peta dan jaringan jalan pada *boundary* pencarian_vcknn menggunakan fungsi view.



Gambar 4.2 *Sequence Diagram* Menampilkan Peta VCKNN

4.3.2 *Sequence Diagram* Menampilkan Peta VN3

Gambar 4.3 merupakan *sequence diagram* Menampilkan Peta VN3 dengan nomor kebutuhan SPR-N-2 yang bertujuan untuk memberikan informasi peta serta jaringan jalan di Kecamatan Lowokwaru. *Sequence diagram* Menampilkan Peta VN3 melibatkan aktor pengguna, *boundary* pencarian_vn3, *controller* Vn3 dan *model* M_jalanv.



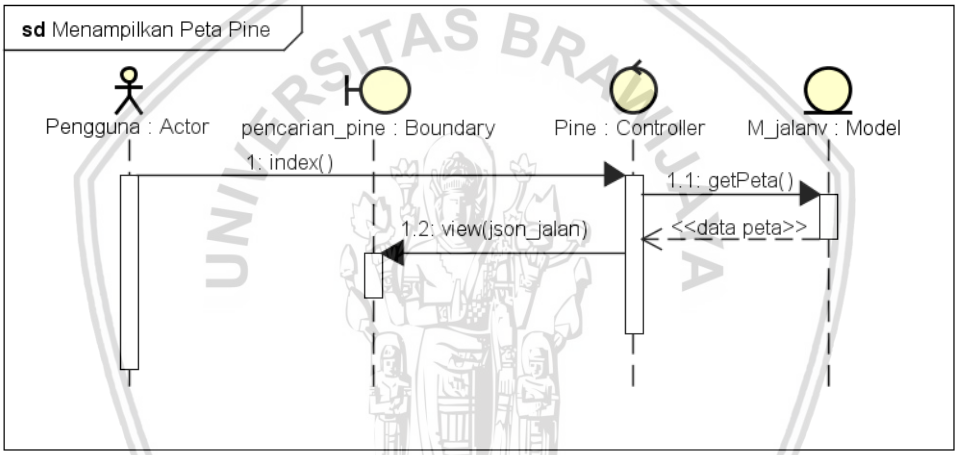
Gambar 4.3 *Sequence Diagram* Menampilkan Peta VN3

Sequence diagram Menampilkan Peta VN3 (Gambar 4.3) dimulai saat pengguna memasukkan *url* sistem pencarian rute paling optimum dimana *url* tersebut akan meneruskan permintaan kepada kelas Vn3, setelah itu dilanjutkan

dengan mengambil data jaringan jalan menggunakan fungsi getPeta pada *model* M_jalanv, lalu menampilkan informasi peta dan jaringan jalan pada *boundary* pencarian_vn3 menggunakan fungsi view.

4.3.3 *Sequence Diagram* Menampilkan Peta PINE

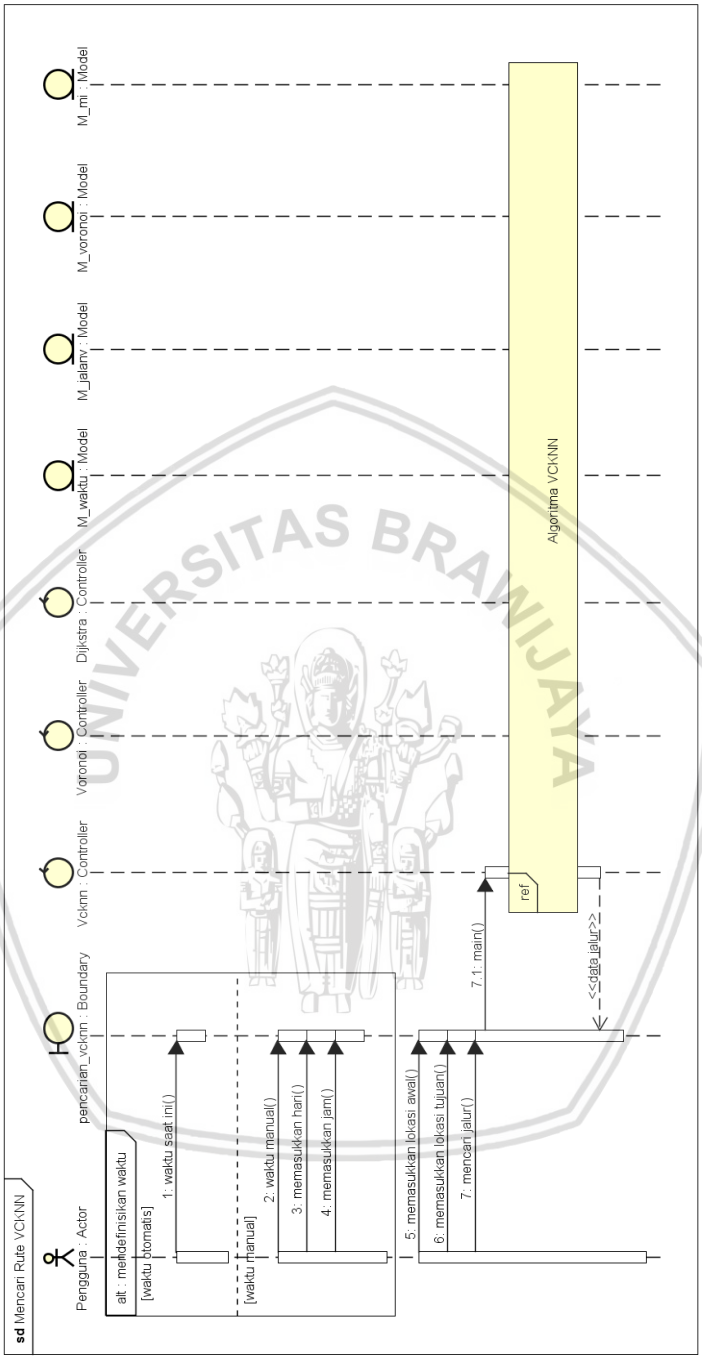
Gambar 4.4 merupakan *sequence diagram* Menampilkan Peta PINE dengan nomor kebutuhan SPR-N-3 yang bertujuan untuk memberikan informasi peta serta jaringan jalan di Kecamatan Lowokwaru. *Sequence diagram* Menampilkan Peta PINE melibatkan aktor pengguna, *boundary* pencarian_pine, *controller* Pine dan *model* M_jalanv. *Sequence diagram* Menampilkan Peta PINE dimulai saat pengguna memasukkan *url* sistem pencarian rute paling optimum dimana *url* tersebut akan meneruskan permintaan kepada kelas Pine, setelah itu dilanjutkan dengan mengambil data jaringan jalan menggunakan fungsi getPeta pada *model* M_jalanv, lalu menampilkan informasi peta dan jaringan jalan pada *boundary* pencarian_pine menggunakan fungsi view.



Gambar 4.4 *Sequence Diagram* Menampilkan Peta PINE

4.3.4 *Sequence Diagram* Mencari Rute VCKNN

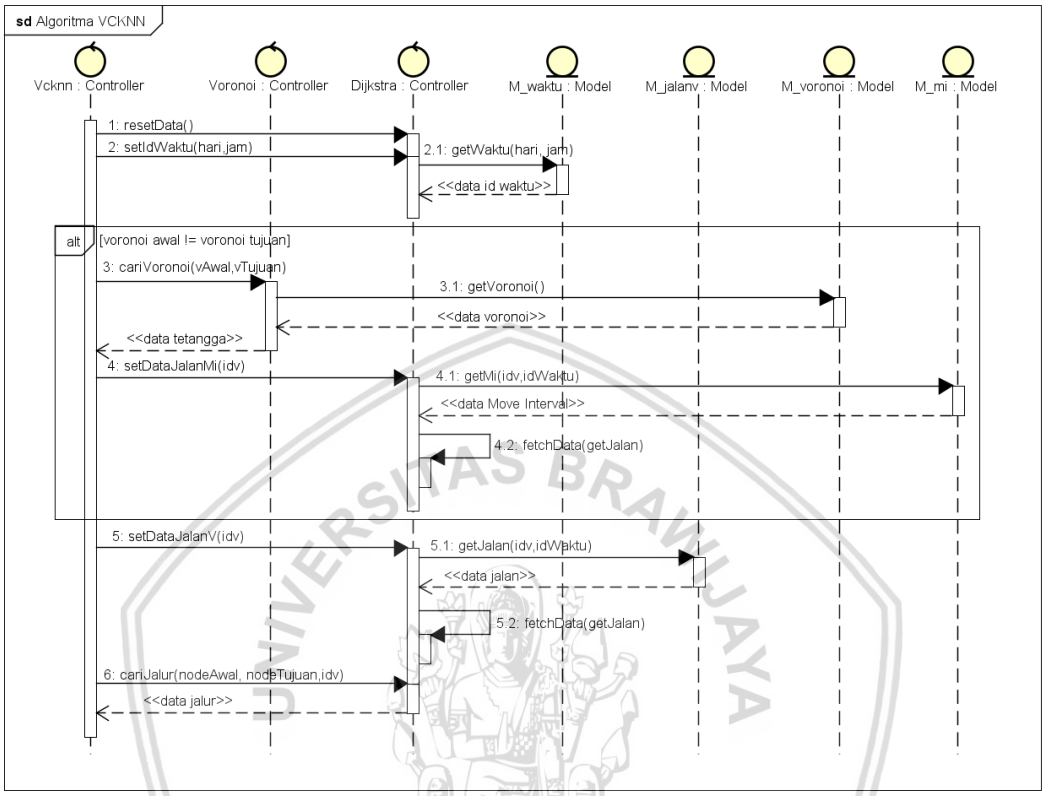
Gambar 4.5 merupakan *sequence diagram* Mencari Rute VCKNN dengan nomor kebutuhan SPR-N-4 yang bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme VCKNN. *Sequence diagram* Mencari Rute VCKNN melibatkan aktor pengguna, *boundary* pencarian_vcknn, *controller* Vcknn, *controller* Voronoi, *controller* Dijkstra, dan *model* M_waktu, *model* M_jalanv, *model* M_voronoi dan *model* M_mi. *Sequence diagram* Mencari Rute VCKNN dimulai dari pengguna mendifinisikan waktu, pada saat mendifinisikan waktu pengguna diberikan pilihan untuk memilih waktu secara otomatis atau secara manual, jika pengguna memilih waktu otomatis maka waktu didefinisikan oleh sistem, jika pengguna memilih waktu manual, maka pengguna akan memasukkan hari dan memasukkan jam, setelah mendefinisikan waktu, pengguna memasukkan lokasi awal serta lokasi tujuan, lalu pengguna menekan tobol cari jalur yang berfungsi untuk meneruskan permintaan kepada kelas Vcknn dan mencari jalur yang dijelaskan dalam *sequence diagram* Algoritme VCKNN (Gambar 4.6).



Gambar 4.5 *Sequence Diagram* Mencari Rute VCKNN

Sequence diagram Algoritme VCKNN (Gambar 4.6) dimulai dari menjalankan fungsi resetData untuk inisialisasi algoritme, setelah itu melakukan inisialisasi waktu dengan fungsi setIdWaktu dimana data id waktu di ambil dari *database*, setelah itu terdapat kondisi jika *voronoi* awal tidak sama dengan *voronoi* tujuan maka akan mencari tetangga terlebih dahulu menggunakan fungsi cariVoronoi serta melakukan inisialisasi data *move interval* berdasarkan *voronoi* tetangga

menggunakan fungsi setDataJalanMi. Setelah itu melakukan inisialisasi data jalan *voronoi* berdasarkan *voronoi* awal dan *voronoi* tujuan menggunakan fungsi setDataJalanV, lalu dilanjutkan dengan mecari jalur menggunakan fungsi cariJalur, data jalur akan dikembalikan kepada *boundary* pencarian_vcknn.



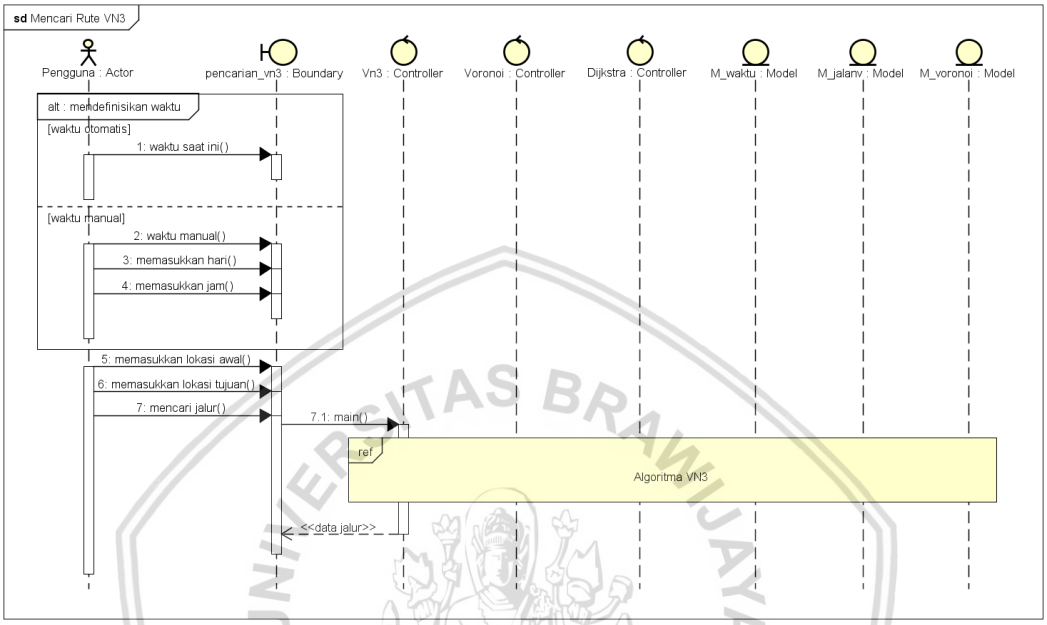
Gambar 4.6 *Sequence Diagram* Algoritme VCKNN

4.3.5 *Sequence Diagram* Mencari Rute VN3

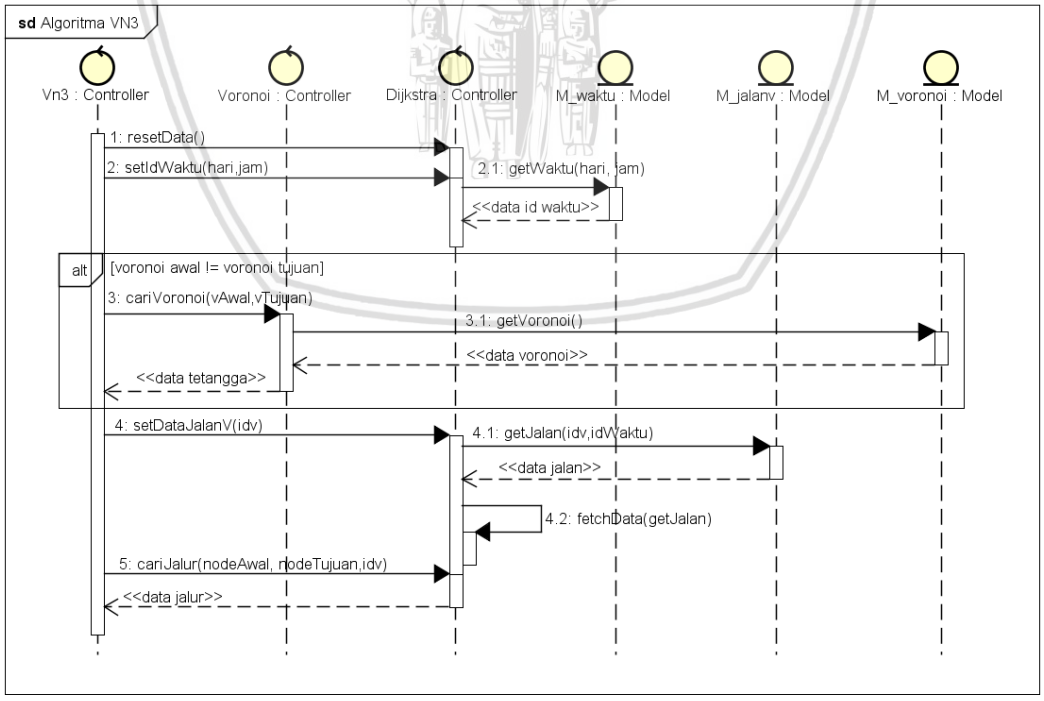
Gambar 4.7 merupakan *sequence diagram* Mencari Rute VN3 dengan nomor kebutuhan SPR-N-5 yang bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme VN3. *Sequence diagram* Mencari Rute VN3 melibatkan aktor pengguna, *boundary* pencarian_vn3, *controller* Vn3, *controller* Voronoi, *controller* Dijkstra, *model* M_waktu, *model* M_jalanv dan *model* M_voronoi. *Sequence diagram* Mencari Rute VN3 dimulai dari pengguna mendifinisikan waktu, pada saat mendifinisikan waktu pengguna diberikan pilihan untuk memilih waktu secara otomatis atau secara manual, jika pengguna memilih waktu otomatis maka waktu didefinisikan oleh sistem, jika pengguna memilih waktu manual, maka pengguna akan memasukkan hari dan memasukkan jam, setelah mendefinisikan waktu, pengguna memasukkan lokasi awal serta lokasi tujuan, lalu pengguna menekan tobol cari jalur yang berfungsi untuk meneruskan permintaan kepada kelas Vn3 dan mencari jalur yang dijelaskan dalam *sequence diagram* Algoritme VN3 (Gambar 4.8).

Sequence diagram Algoritme VN3 (Gambar 4.8) dimulai dari menjalankan fungsi resetData untuk inisialisasi algoritme, setelah itu melakukan inisialisasi waktu dengan fungsi setIdWaktu dimana data id waktu diambil dari *database*,

setelah itu terdapat kondisi jika *voronoi* awal tidak sama dengan *voronoi* tujuan maka akan mencari tetangga terlebih dahulu menggunakan fungsi cariVoronoi. Setelah itu melakukan inisialisasi data jalan *voronoi* berdasarkan *voronoi* awal, *voronoi* tujuan dan *voronoi* tetangga menggunakan fungsi setDataJalanV, lalu dilanjutkan dengan mecari jalur menggunakan fungsi cariJalur, data jalur akan dikembalikan kepada *boundary* pencarian_vn3.

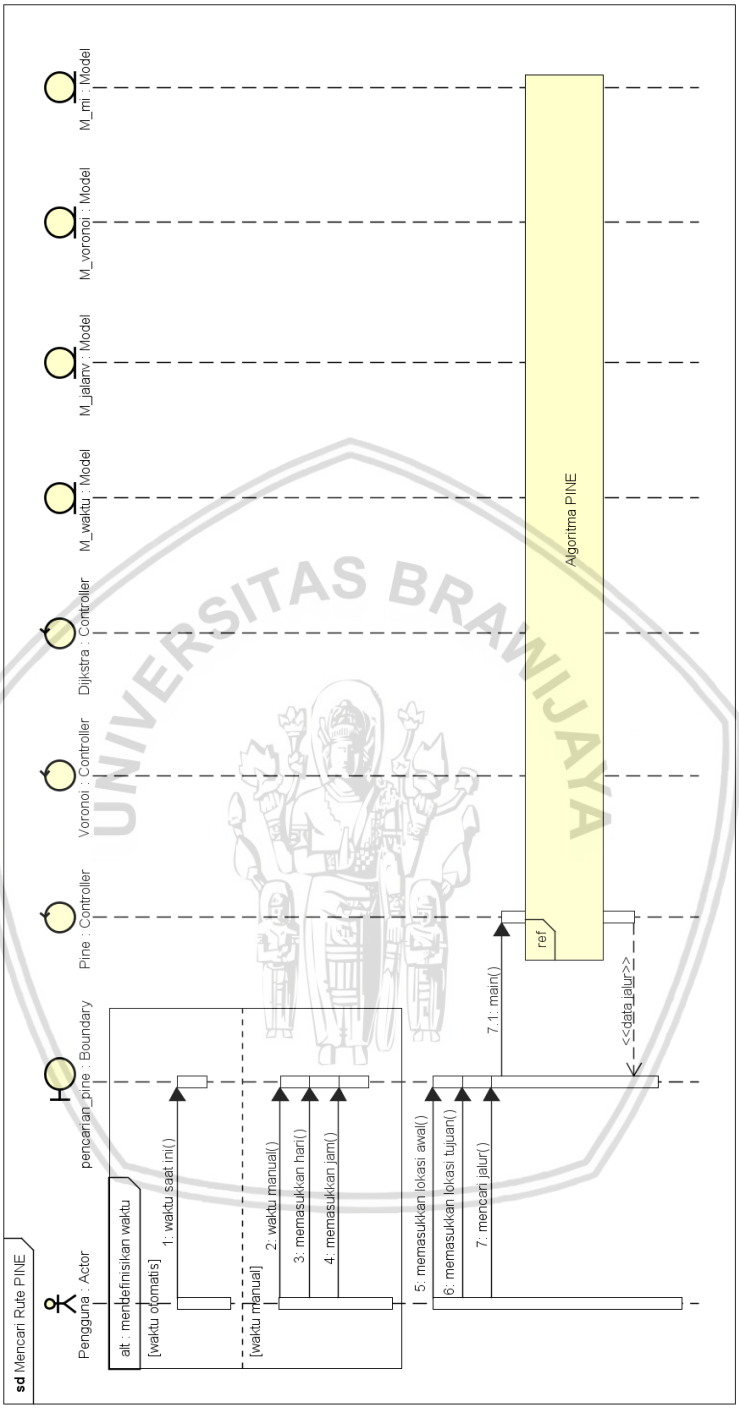


Gambar 4.7 *Sequence Diagram* Mencari Rute VN3



Gambar 4.8 *Sequence Diagram* Algoritme VN3

4.3.6 *Sequence Diagram* Mencari Rute PINE

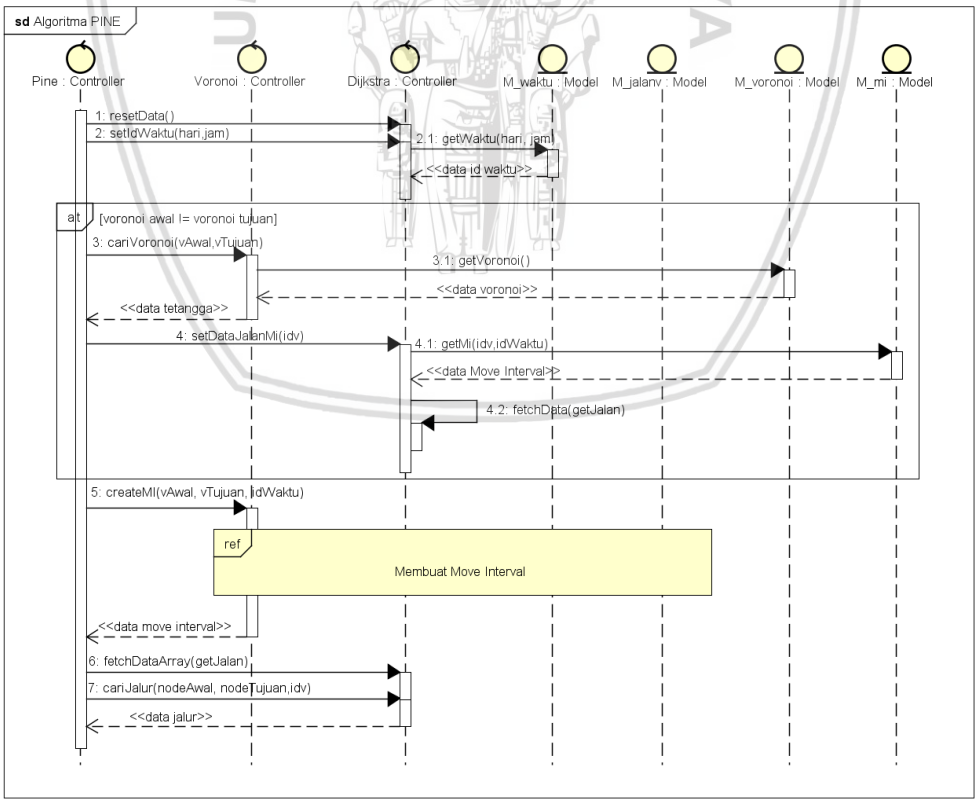


Gambar 4.9 *Sequence Diagram* Mencari Rute PINE

Gambar 4.9 merupakan *sequence diagram* Mencari Rute PINE dengan nomor kebutuhan SPR-N-6 yang bertujuan untuk memberikan rekomendasi rute paling optimum menggunakan algoritme PINE. *Sequence diagram* Mencari Rute PINE melibatkan aktor pengguna, *boundary* pencarian_pine, *controller* Pine, *controller* Voronoi, *controller* Dijkstra, *model* M_waktu, *model* M_jalanv, *model* M_voronoi dan *model* M_mi. *Sequence diagram* Mencari Rute PINE dimulai dari pengguna

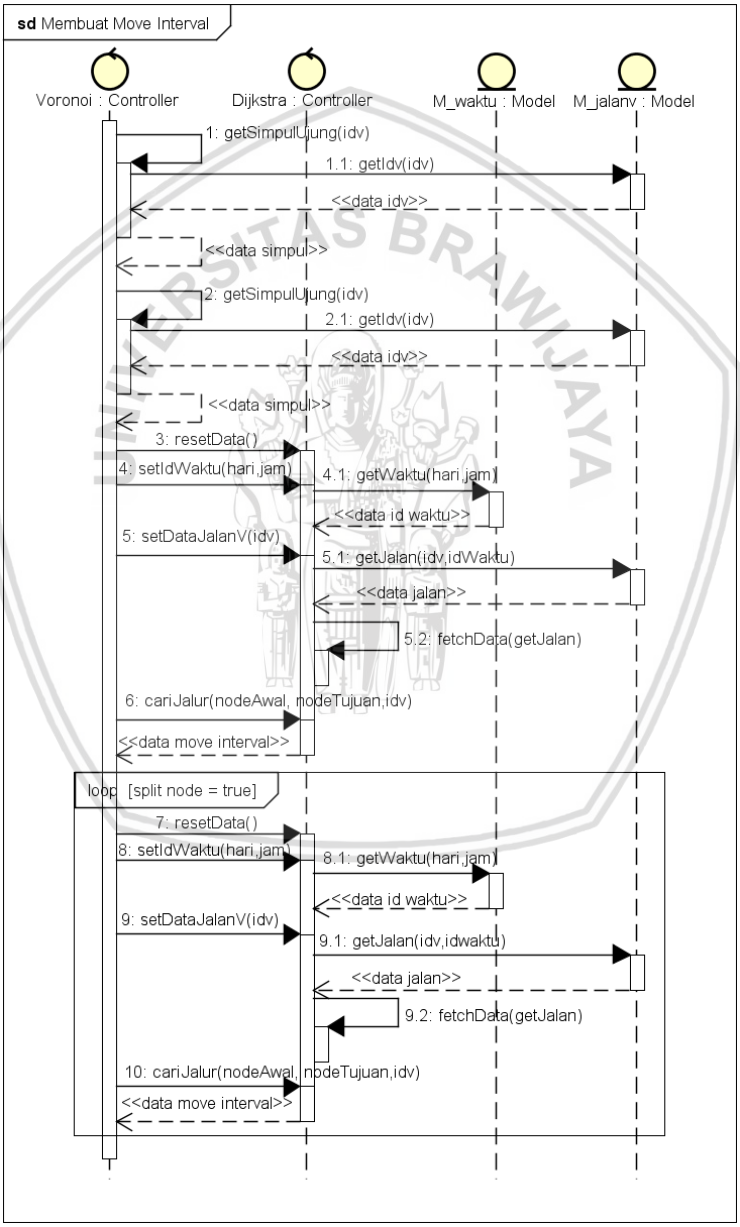
mendifinisikan waktu, pada saat mendifinisikan waktu pengguna diberikan pilihan untuk memilih waktu secara otomatis atau secara manual, jika pengguna memilih waktu otomatis maka waktu didefinisikan oleh sistem, jika pengguna memilih waktu manual, maka pengguna akan memasukkan hari dan memasukkan jam, setelah mendefinisikan waktu, pengguna memasukkan lokasi awal serta lokasi tujuan, lalu pengguna menekan tobol cari jalur yang berfungsi untuk meneruskan permintaan kepada kelas Pine dan mencari jalur yang dijelaskan dalam *sequence diagram* Algoritme PINE (Gambar 4.10).

Sequence diagram Algoritme PINE (Gambar 4.10) dimulai dari menjalankan fungsi resetData untuk inisialisasi algoritme, setelah itu melakukan inisialisasi waktu dengan fungsi setIdWaktu dimana data id waktu di ambil dari *database*, setelah itu terdapat kondisi jika *voronoi* awal tidak sama dengan *voronoi* tujuan maka akan mencari tetangga terlebih dahulu menggunakan fungsi cariVoronoi serta melakukan inisialisasi data *move interval* berdasarkan *voronoi* tetangga menggunakan fungsi setDataJalanMi. Setelah itu membuat *move interval* pada *voronoi* awal dan *voronoi* tujuan yang dijelaskan dalam *sequence diagram* Membuat *Move Interval* (Gambar 4.11), lalu setelah mendapatkan *move interval* pada *voronoi* awal dan *voronoi* tujuan, selanjutnya melakukan inisilalisasi data *move interval* berdasarkan *voronoi* awal dan *voronoi* tujuan menggunakan fungsi fetchDataArray. Setelah itu dilanjutkan dengan mecari jalur menggunakan fungsi cariJalur, data jalur akan dikembalikan kepada *boundary* pencarian_pine.



Gambar 4.10 *Sequence Diagram* Algoritme PINE

Pada *sequence diagram* Membuat *Move Interval* (Gambar 4.11) dimulai dengan mencari *split node* pada *voronoi* awal dan *voronoi* tujuan menggunakan fungsi getSimpulUjung, lalu mencari *move interval* pada *voronoi* awal dengan menggunakan fungsi resetData untuk inisialisasi algoritme, fungsi setIdWaktu untuk inisalisasi waktu, fungsi setDataJalanV untuk insialisasi data jalan pada *voronoi* awal dan fungsi cariJalur untuk mencari *move interval* pada *voronoi* awal, setelah itu dilanjutkan dengan mencari *move interval* pada *voronoi* tujuan dengan menggunakan fungsi resetData untuk inisialisasi algoritme, fungsi setIdWaktu untuk inisalisasi waktu, fungsi setDataJalanV untuk insialisasi data jalan pada *voronoi* tujuan dan fungsi cariJalur pada seluruh *split node* di *voronoi* tujuan.



Gambar 4.11 *Sequence Diagram* Membuat *Move Interval*

4.4 Perancangan *Class Diagram*

Pada tahap perancangan *class diagram*, berisi mengenai struktur dari sistem dengan mendefinisikan kelas-kelas yang nantinya digunakan untuk mengembangkan sistem pencarian rute paling optimum. Perancangan *class diagram* diawali dengan membuat kelas diagram analisis, diagram kelas analisis menjadi acuan dalam pembuatan kelas diagram domain *model*, setelah itu membuat kelas diagram pada domain model dan kelas diagram pada domain *cotroller*.

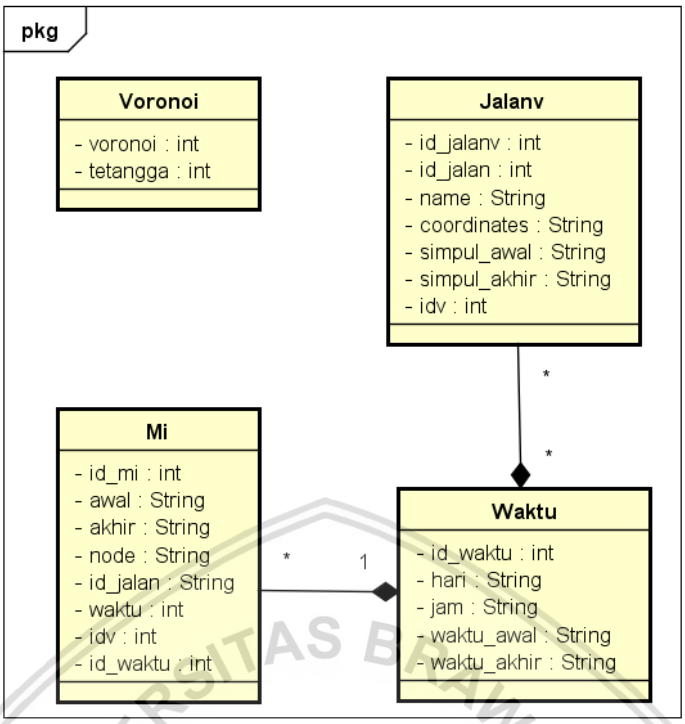
4.4.1 Diagram Kelas Analisis

Gambar 4.12 menunjukkan diagram kelas analisis pada sistem pencarian rute paling optimum. Kelas diagram analisis didapatkan dengan memilih serta mengelompokan kosakata yang diidentifikasi pada tahap analisis persyaratan. Dalam sistem pencarian rute paling optimum teridentifikasi 4 kelas beserta atributnya, diagram kelas analisis akan menjadi acuan dalam pembuatan kelas diagram domain *model* dan perancangan basis data. Kelas Voronoi merupakan representasi dari objek poligon *voronoi*, setiap poligon *voronoi* berbatasan dengan *voronoi* lainnya yang disebut dengan tetangga. Kelas Voronoi memiliki atribut voronoi berisi nilai *vorinoi* dan atribut tertangga berisi nilai tetanganya.

Kelas Jalanv merupakan representasi dari objek jalan *voronoi*, jalan *voronoi* merupakan jalan yang sudah dikelompokkan berdasarkan poligon *voronoi*. Kelas Jalanv memiliki atribut id_jalanv berisi nilai id jalan yang sudah dikelompokkan, id_jalan berisi nilai id jalan sebelum dikelompokkan, name berisi nama jalan, coordinates berisi titik-titik koordinat untuk membuat garis pada jalan, simpul_awal berisi titik koordinat awal pada jalan, simpul_akhir berisi titik koordinat akhir pada jalan dan idv berisi nilai id voronoi jalan.

Kelas Waktu merupakan representasi dari objek waktu, data waktu dikelompokkan berdasarkan waktu terjadinya kemacetan yang berpengaruh kepada waktu tempuh pada jalan. Kelas Waktu memiliki atribut id_waktu berisi nilai id waktu, hari berisi data hari, jam berisi data jam, waktu_awal berisi data waktu awal, waktu_akhir berisi data waktu akhir.

Kelas Mi merupakan representasi dari objek *move interval*, *move iterval* adalah segmen baru yang yang di bentuk dari 2 titik yang berbatasan pada poligon *voronoi*. Kelas Mi memiliki atribut id_mi berisi nilai id mi, awal berisi titik koordinat awal pada *move interval*, akhir berisi titik koordinat akhir pada *move interval*, *node* berisi *node-node* yang dilalui, id_jalan berisi jalan-jalan yang dilalui, waktu berisi nilai waktu tempuh *move interval*, idv berisi nilai id voronoi *move interval*, id_waktu berisi nilai id waktu.



Gambar 4.12 Kelas Diagram Analisis

4.4.2 Diagram Kelas Perancangan

Setelah melakukan tahap diagram kelas analisis, dilanjutkan dengan tahapan pembuatan kelas diagram pada domain *model* dan kelas diagram pada domain *cortroller*, pembuatan kelas diagram perancangan mengacu pada kelas diagram analisis. Pada sistem pencarian rute paling optimum menggunakan kerangka kerja *Codeigniter* sehingga kelas diagram serta *sequence diagram* menerapkan pola *Model View Controller* dan setiap *model* memiliki relasi dengan CI_Model serta setiap *controller* memiliki relasi dengan CI_Controller

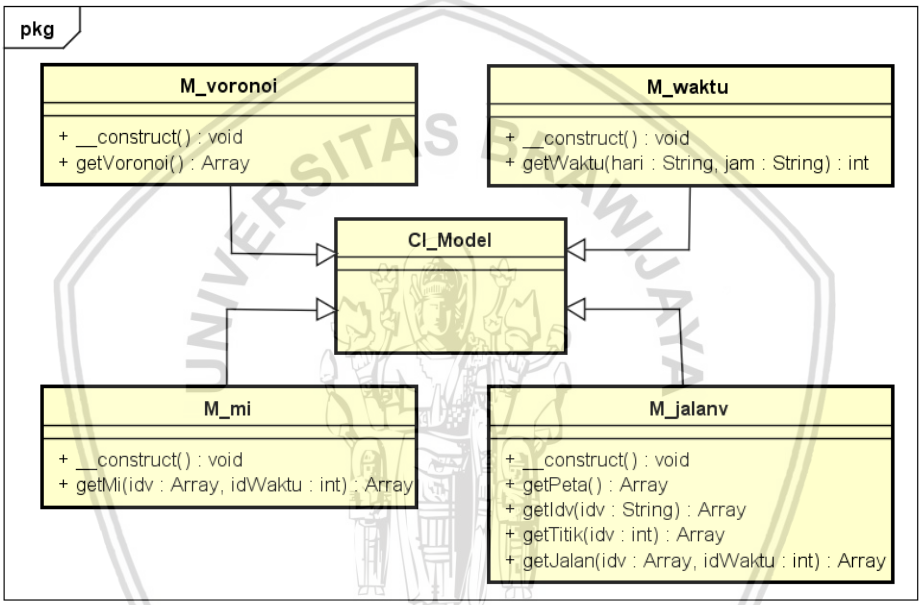
Terdapat 9 kelas diagram perancangan dalam sistem pencarian rute paling optimum, 4 kelas sebagai kelas diagram domain *model* (Gambar 4.13) dan 5 kelas sebagai kelas diagram domain *controller* (Gambar 4.14). Hasil dari tahapan pemodelan kelas diargram perancangan digunakan sebagai panduan dalam mengembangkan sistem pencarian rute paling optimum.

Pada kelas diagram domain model terdapat 4 kelas, yaitu kelas M_voronoi, M_waktu, M_mi, dan M_jalanv (Gambar 4.13). Pada kelas M_voronoi memiliki 2 fungsi, yaitu fungsi __construct tanpa pengembalian nilai yang digunakan untuk melakukan inisialisasi kelas dan fungsi getVoronoi dengan pengembalian data *voronoi* yang digunakan untuk mengambil data *voronoi* beserta tetangganya.

Pada kelas M_waktu memiliki 2 fungsi yaitu fungsi __construct tanpa pengembalian nilai yang digunakan untuk melakukan inisialisasi kelas dan fungsi getWaktu dengan pengembalian nilai id waktu yang digunakan untuk mengambil nilai id waktu berdasarkan hari dan jam. Pada kelas M_mi memiliki 2 fungsi yaitu fungsi __construct tanpa pengembalian nilai yang digunakan untuk melakukan

inisialisasi kelas dan fungsi getMi dengan pengembalian data *move interval* yang digunakan untuk mengambil data *move interval* berdasarkan id waktu dan id voronoi.

Pada kelas M_jalanv memiliki 4 fungsi, yaitu fungsi __construct tanpa pengembalian nilai yang digunakan untuk melakukan inisialisasi kelas, fungsi getPeta dengan pengembalian data jaringan jalan yang digunakan untuk mengambil data jaringan jalan di Kecamatan Lowokwaru, fungsi getIdv dengan pengembalian nilai id voronoi yang digunakan untuk mengambi data *voronoi* berdasarkan id voronoi, fungsi getTitik dengan pengembalian data titik *split node* yang digunakan untuk mengambil data *split node* berdasarkan id voronoi dan fungsi getJalan dengan pengembalian data jaringan jalan yang digunakan untuk mengambil data jaringan jalan berdasarkan id voronoi dan id waktu.



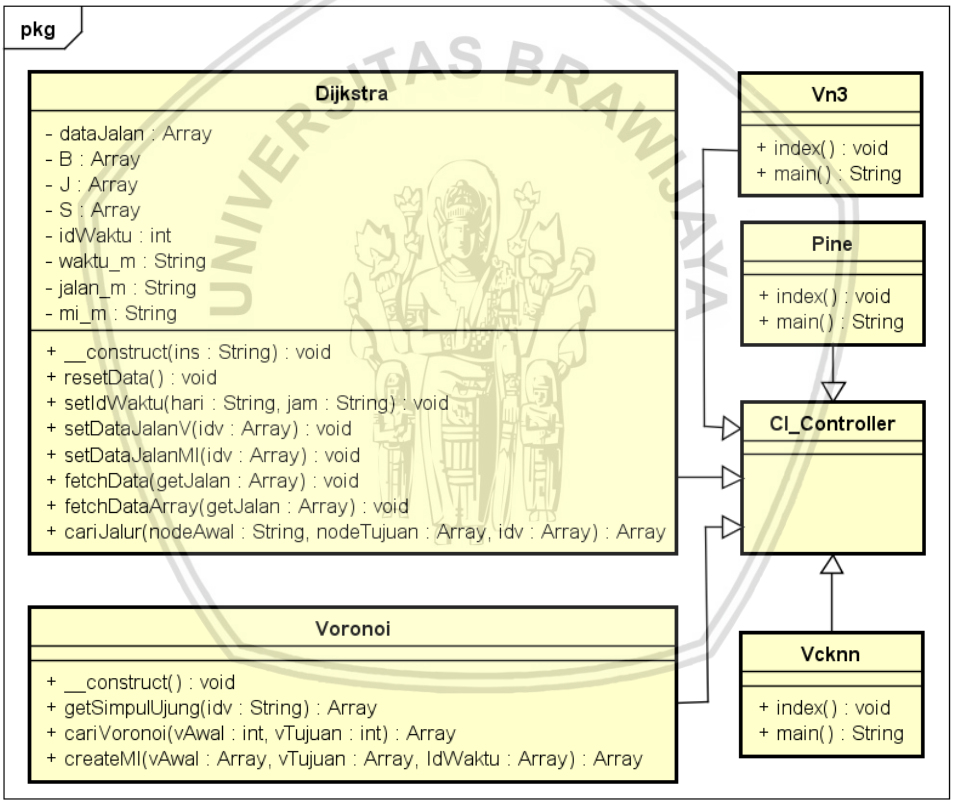
Gambar 4.13 Kelas Diagram Domain *Model*.

Pada kelas diagram domain *controller* terdapat 5 kelas, yaitu kelas Dijkstra, kelas Voronoi, kelas Vcknn, kelas Pine dan kelas Vn3 (Gambar 4.14). Pada kelas Dijkstra memiliki 8 atribut yaitu atribut dataJalan berisi data jalan yang akan digunakan dalam pencarian jalur, atribut B berisi data *node* yang belum dikunjungi, atribut J berisi data jalur, atribut S berisi data *node* yang sudah dikunjungi, atribut idWaktu berisi nilai id waktu, atribut waktu_m berisi variabel untuk *model* waktu, atribut jalan_m berisi variabel untuk *model* jalan dan atribut mi_m berisi variabel untuk *model* mi.

Terdapat 4 fungsi dalam kelas Dijkstra, yaitu fungsi __construct tanpa pengembalian nilai yang digunakan untuk melakukan inisialisasi kelas, fungsi resetData tanpa pengembalian nilai yang digunakan untuk inisialisasi algoritme, fungsi setIdWaktu tanpa pengembalian nilai yang digunakan untuk inisialisasi waktu berdasarkan hari dan jam, fungsi setDataJalanV tanpa pengembalian nilai yang digunakan untuk inisialisasi data jalan *voronoi* berdasarkan id voronoi, fungsi setDataJalanMi tanpa pengembalian nilai yang digunakan untuk inisialisasi data

move interval, fungsi fetchData tanpa pengembalian nilai yang digunakan untuk inisialisasi data jaringan jalan berdasarkan id voronoi, fungsi fetchDataArray tanpa pengembalian nilai yang digunakan untuk inisialisasi jaringan jalan berdasarkan id voronoi dan fungsi cariJalur dengan pengembalian data jalur yang digunakan untuk mencari jalur dengan bobot terkecil berdasarkan *node* awal, *node* tujuan dan id voronoi.

Pada kelas Voronoi terdapat 4 fungsi, yaitu fungsi __construct tanpa pengembalian nilai yang digunakan untuk melakukan inisialisasi kelas, fungsi getSimpulUjung dengan pengembalian data *split node* yang digunakan untuk mencari *split node* berdasarkan id voronoi, fungsi cariVoronoi dengan pengembalian data *voronoi* yang digunakan untuk mencari tetangga berdasarkan id voronoi, fungsi createMI dengan pengembalian nilai data *move interval* yang digunakan untuk mencari *move interval* berdasarkan *voronoi* awal, *voronoi* tujuan dan id waktu.

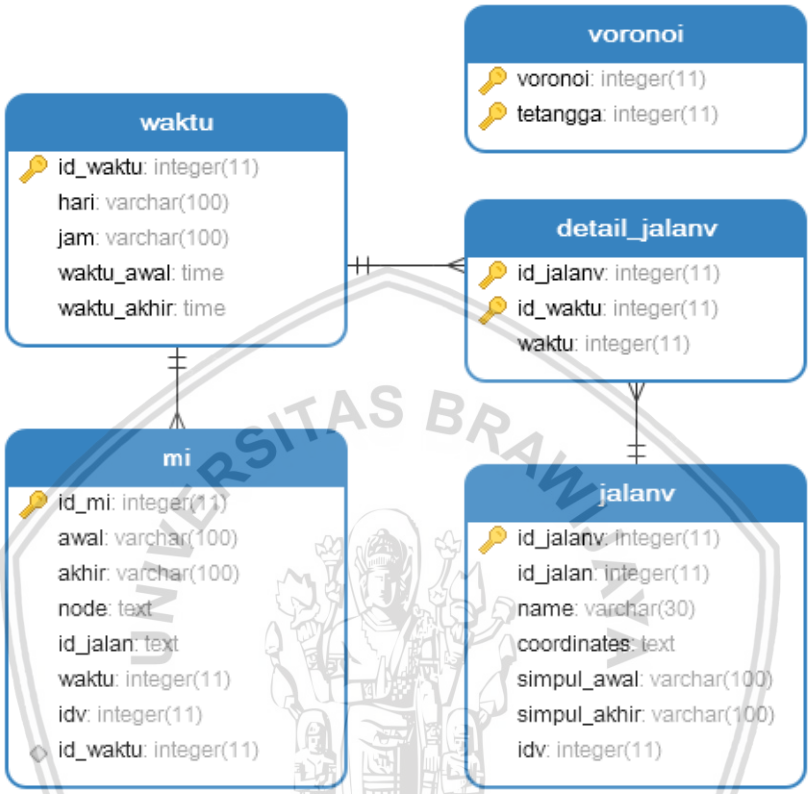


Gambar 4.14 Kelas Diagram Domain *Controller.*

Pada kelas Vcknn terdapat 2 fungsi yaitu fungsi index tanpa pengembalian nilai yang digunakan untuk menampilkan peta dan fungsi main dengan pengembalian data rute paling optimum yang digunakan untuk mencari rute paling optimum menggunakan algoritme VCKNN. Pada kelas Pine terdapat 2 fungsi yaitu fungsi index tanpa pengembalian nilai yang digunakan untuk menampilkan peta dan fungsi main dengan pengembalian data rute paling optimum yang digunakan untuk mencari rute paling optimum menggunakan algoritme PINE. Pada kelas Vn3 terdapat 2 fungsi yaitu fungsi index tanpa pengembalian nilai yang digunakan

untuk menampilkan peta dan fungsi main dengan pengembalian data rute paling optimum yang digunakan untuk mencari rute paling optimum menggunakan algoritme VN3.

4.5 Perancangan *Data Model*



Gambar 4.15 Perancangan *Data Model*

Gambar 4.15 menjelaskan perancangan *data model* pada sistem pencarian rute paling optimum, perancangan *data model* dibuat berdasarkan pemodelan kelas diagram analisis (Gambar 4.15) dalam bentuk yang sudah dinormalisasi, normalisasi dilakukan pada relasi antara tabel jalanv dan waktu karena terdapat relasi banyak ke banyak, sehingga dibuat tabel baru yaitu tabel detail_jalanv. Pada sistem pencarian rute paling optimum terdapat 5 tabel yaitu tabel voronoi untuk menyimpan data *voronoi* dan tetangganya, tabel mi untuk menyimpan data *move interval*, tabel jalanv untuk menyimpan data jalan *voronoi*, tabel waktu untuk meyimpan data waktu dan tabel detail_jalanv untuk menyimpan waktu tempuh berdasarkan jalan dan waktu.

4.5.1 Tabel Voronoi

Pada tabel voronoi menyimpan data *voronoi* beserta tetangganya, terdapat 2 kolom dalam tabel voronoi yaitu voronoi dan tetangga. *Primary key* pada tabel voronoi adalah voronoi dan tetangga. Tabel 4.17 merupakan penjelasan kolom-kolom dari tabel voronoi.

Tabel 4.17 Tabel Voronoi

Nomor	Nama Kolom	Tipe Data	Panjang	Keterangan
1	Voronoi	Integer	11	Nomor *voronoi*
2	Tetangga	Integer	11	Tetangga *voronoi*

4.5.2 Tabel Waktu

Pada tabel waktu menyimpan data waktu yang sudah dikelompokkan berdasarkan waktu kemacetan, terdapat 5 kolom dalam tabel waktu yaitu id_waktu, hari, jam, waktu_awal dan waktu_akhir. *Primary key* pada tabel waktu adalah id_waktu. Tabel 4.18 merupakan penjelasan kolom-kolom dari tabel waktu.

Tabel 4.18 Tabel Waktu

Nomor	Nama Kolom	Tipe Data	Panjang	Keterangan
1	Id_waktu	Integer	11	Id waktu
2	Hari	Varchar	100	Hari
3	Jam	Varchar	100	Jam
4	Waktu_awal	Time	-	Batas waktu awal
5	Waktu_akhir	Time	-	Batas waktu akhir

4.5.3 Tabel Mi

Pada tabel mi menyimpan data *move interval*, terdapat 8 kolom dalam tabel mi yaitu id_mi, awal, akhir, *node*, id_jalan, waktu, idv dan id_waktu. *Primary key* pada tabel mi adalah id_mi, sedangkan yang menjadi *foreign key* adalah id_waktu yang mengacu kolom id_waktu pada tabel id_waktu. Tabel 4.19 merupakan penjelasan kolom-kolom dari tabel mi.

Tabel 4.19 Tabel Mi

Nomor	Nama Kolom	Tipe Data	Panjang	Keterangan
1	Id_mi	Integer	11	Id *move interval*
2	Awal	Varchar	100	Simpul awal *move interval*
3	Akhir	Varchar	100	Simpul akhir *move interval*
4	Node	Text	-	Simpul-Simpul yang dilewati
5	Id_jalan	Text	-	Id_jalan-id_jalan yang dilewati
6	Waktu	Integer	11	Waktu tempuh *move interval*
7	Idv	Integer	11	Id voronoi *move interval*
8	Id_waktu	Integer	11	Id waktu *move interval*

4.5.4 Tabel Jalanv

Pada tabel jalanv menyimpan data jaringan jalan *voronoi* pada Kecamatan Lowokwaru, terdapat 7 kolom dalam tabel jalanv yaitu id_jalanv, id_jalan, name, coordinates, simpul_awal, simpul_akhir dan idv. *Primary key* pada tabel jalanv adalah id_jalanv. Tabel 4.18 merupakan penjelasan kolom-kolom dari tabel jalanv.

Tabel 4.20 Tabel Jalanv

Nomor	Nama Kolom	Tipe Data	Panjang	Keterangan
1	Id_jalanv	Integer	11	Id jalan *voronoi*
2	Id_jalan	Integer	11	Id jalan
3	Name	Varchar	30	Nama jalan
4	Coordinates	Text	-	Koordinat jalan
5	Simpul_awal	Varchar	100	Simpul awal jalan *voronoi*
6	Simpul_akhir	Varchar	100	Simpul akhir jalan *voronoi*
7	Idv	Integer	11	Id voronoi jalan *voronoi*

4.5.5 Tabel Detail_jalanv

Pada tabel detail_jalanv menyimpan waktu tempuh berdasarkan jalan dan waktu, terdapat 3 kolom dalam tabel detail_jalanv yaitu id_jalanv, id_waktu dan waktu. Tabel detail_jalanv memiliki 2 *foreign key*, yaitu id_jalanv yang mengacu kolom id_jalanv pada tabel jalanv dan id_waktu yang mengacu kolom id_waktu pada tabel waktu, yang menjadi *primary key* pada tabel detail_jalanv adalah id_jalan dan id_waktu. Tabel 4.21 merupakan penjelasan kolom-kolom dari tabel detail_jalanv.

Tabel 4.21 Tabel Detail_jalanv

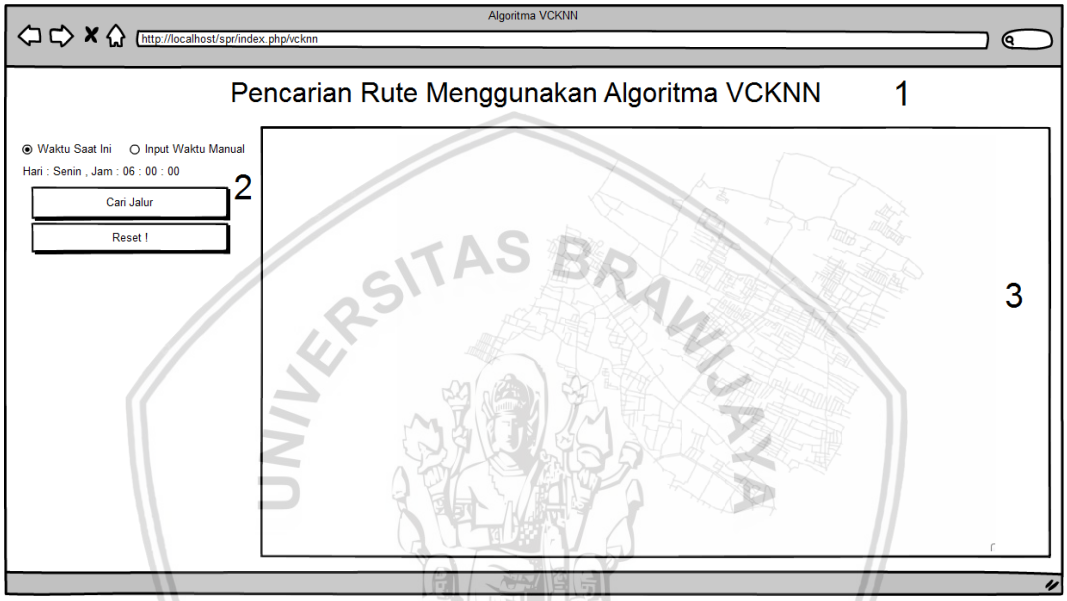
Nomor	Nama Kolom	Tipe Data	Panjang	Keterangan
1	Id_jalanv	Integer	11	Id jalan *voronoi*
2	Id_waktu	Integer	11	Id waktu
3	Waktu	Integer	11	Waktu tempuh jalan

4.6 Perancangan Antarmuka Pengguna

Pada tahapan perancangan antarmuka pengguna, akan dijelaskan perancangan antarmuka dari sistem pencarian rute paling optimum yaitu perancangan antarmuka menampilkan peta dan perancangan antarmuka pencarian rute.

4.6.1 Antarmuka Menampilkan Peta

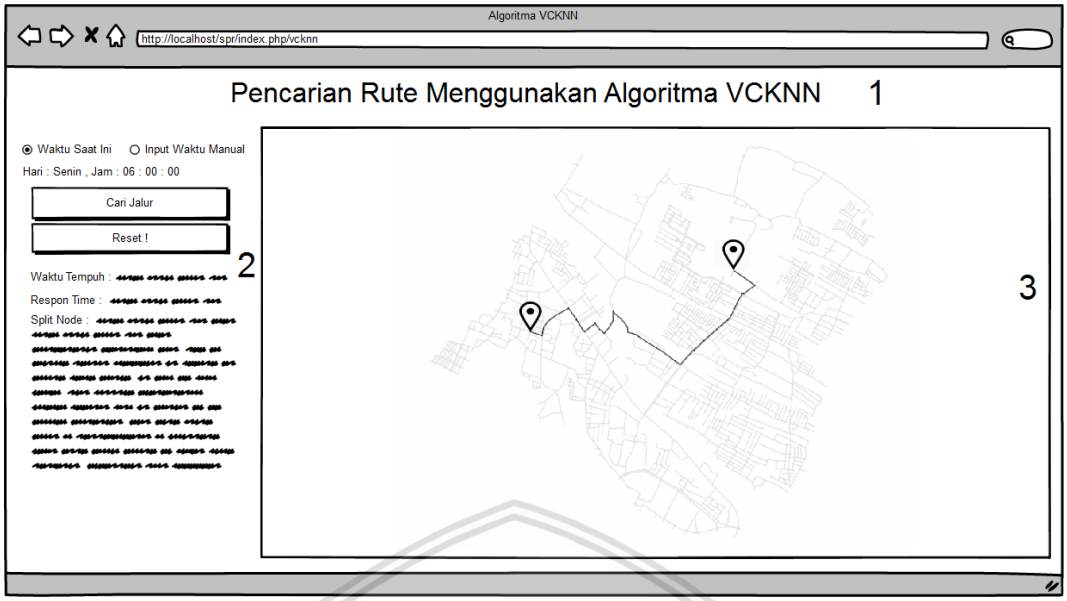
Gambar 4.16 merupakan tampilan antarmuka menampilkan peta, antarmuka menampilkan peta dilihat oleh pengguna ketika pengguna ingin melihat peta dan jaringan jalan di Kecamatan Lowokwaru. Pada bagian 1 (Gambar 4.16) terdapat judul, judul memberikan informasi tetang algoritme yang digunakan dalam pencarian rute paling optimum. Pada bagian 2 (Gambar 4.16) terdapat navigasi, navigasi berisi pilihan untuk mendefinisikan waktu, tombol untuk mencari jalur dan tombol untuk melakukan reset peta. Pada bagian 3 (Gambar 4.16) terdapat *WebGIS*, *WebGIS* berisi peta serta jaringan jalan Kecamatan Lowokwaru.



Gambar 4.16 Perancangan Antarmuka Menampilkan Peta

4.6.2 Antarmuka Pencarian Rute

Gambar 4.17 merupakan tampilan antarmuka pencarian rute, antarmuka pencarian rute dilihat oleh pengguna ketika pengguna ingin mencari rute paling optimum. Pada bagian 1 (Gambar 4.17) terdapat judul, judul memberikan informasi tetang algoritme yang digunakan dalam pencarian rute paling optimum. Pada bagian 2 (Gambar 4.17) terdapat navigasi, navigasi berisi pilihan untuk mendefinisikan waktu, tombol ontuk mencari jalur, tombol untuk melakukan reset peta dan keterangan dari hasil pencarian rute seperti waktu tempuh, *response time* serta *split node, split node* adalah *node* yang dilalui dalam rute yang direkomendasikan. Pada bagian 3 (Gambar 4.17) terdapat *WebGIS*, *WebGIS* berisi peta, jaringan jalan Kecamatan Lowokwaru dan rute paling optimum berdasarkan lokasi awal dan lokasi tujuan.



Gambar 4.17 Perancangan Antarmuka Pencarian Rute.

4.7 Perancangan Pengujian

Pada tahapan perancangan pengujian, akan dijelaskan skenario dan kasus uji yang dilakukan pada tahapan pengujian sistem. Pada sistem pencarian rute paling optimum dilakukan pengujian performa pada aspek *response time* dan pengujian statistik *Chi-square*. Pengujian performa dilakukan untuk mendapatkan nilai *response time* dan jumlah *node* yang dialui setiap skenario pada masing-masing algoritme, pengujian statistik *Chi-square* dilakukan untuk melihat perbandingan performa pada setiap algoritme.

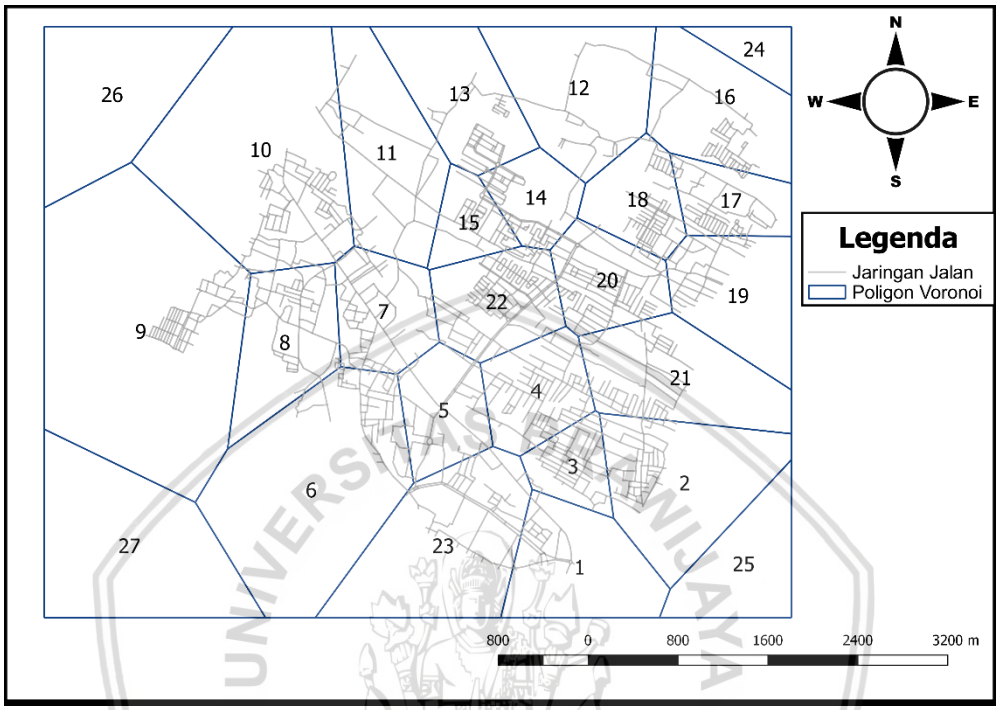
4.7.1 *Performance Testing*

Pada sistem pencarian rute paling optimum dilakukan pengujian performa pada aspek *response time*, *response time* adalah waktu yang dibutuhkan sistem untuk menanggapi permintaan pengguna (Molyneaux, 2015). *Response time* dapat diketahui dengan cara mencari selisih waktu antara pengguna mengirim permintaan mencari jalur dan pengguna menerima hasil rekomendasi jalur. Tabel 4.22 merupakan kode program dalam bahasa pemrograman *javascript* untuk mendapatkan *response time* pada sistem pencarian rute paling optimum. Pada baris 1 (Tabel 4.22) digunakan untuk mendapatkan waktu pada saat pengguna mengirim permintaan untuk mencari jalur, baris 2 (Tabel 4.22) digunakan untuk mendapatkan waktu pada saat pengguna menerima hasil rekomendasi jalur, baris 3 (Tabel 4.22) digunakan untuk mendapatkan *response time* dalam satuan detik.

Tabel 4.22 Kode Program Mendapatkan *Response time*

No	Kode Program
1	`var permintaan = (new Date()).getTime();`
2	`var tanggapan = (new Date()).getTime();`
3	`var responTime = (tanggapan - permintaan)/1000;`

Pengujian *response time* sistem pencarian rute paling optimum dibagi menjadi 3 skenario, Tabel 4.23 menjelaskan skenario dari pengujian *response time*. Pembagian skenario berdasarkan dari level tetangga *voronoi* dari setiap id voronoi (Gambar 4.18), yaitu level 1 tetangga *voronoi*, level 3 tetangga *voronoi* dan level 6 tetangga *voronoi*.



Gambar 4.18 Pembagian Poligon *Voronoi*

Tabel 4.23 Skenario Pengujian *Response time*

Nomor Skenario	Level Tetangga	Lokasi Awal	*Voronoi* Awal	Lokasi Tujuan	*Voronoi* Tujuan
Skenario 1	1	[112.593124,-7.943738]	9	[112.60308,-7.946248001]	8
Skenario 2	3	[112.593124,-7.943738]	9	[112.623641,-7.949381]	4
Skenario 3	6	[112.593124,-7.943738]	9	[112.640892,-7.933180000999 99]	17

Untuk mendapatkan nilai *response time* dan *node* yang dilalui pada masing-masing algoritme, dilakukan pengujian performa dengan 3 skenario (Tabel 4.23) pada masing-masing algoritme. Setiap pengujian performa diberikan kode sebagai identitas, keterangan kodefikasi pengujian performa sebagai berikut.

Kode Kebutuhan Pengguna: SPR-PT-xx

Keterangan : SPR : Singkatan dari Sistem Pencarian Rute

PT : Kode representasi pengujian performa

XX : Nomor urut pengujian performa

4.7.1.1 Pengujian Performa Skenario 1 Algoritme VCKNN

Tabel 4.24 merupakan kasus uji pengujian performa pada skenario 1 menggunakan algoritme VCKNN. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VCKNN. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VCKNN.

Tabel 4.24 Kasus Uji Pengujian Performa Skenario 1 Algoritme VCKNN

Kode Pengujian	SPR-PT-01	
Nomor Skenario	Skenario 1	
Algoritme	VCKNN	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VCKNN.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.60308,-7.946248001]
	Voronoi Tujuan	8
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VCKNN. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

4.7.1.2 Pengujian Performa Skenario 2 Algoritme VCKNN

Tabel 4.25 merupakan kasus uji pengujian performa pada skenario 2 menggunakan algoritme VCKNN. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan

algoritme VCKNN. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VCKNN.

Tabel 4.25 Kasus Uji Pengujian Performa Skenario 2 Algoritme VCKNN

Kode Pengujian	SPR-PT-02	
Nomor Skenario	Skenario 2	
Algoritme	VCKNN	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VCKNN.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.623641,-7.949381]
	Voronoi Tujuan	4
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VCKNN. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

4.7.1.3 Pengujian Performa Skenario 3 Algoritme VCKNN

Tabel 4.26 merupakan kasus uji pengujian performa pada skenario 3 menggunakan algoritme VCKNN. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VCKNN. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VCKNN.

Tabel 4.26 Kasus Uji Pengujian Performa Skenario 3 Algoritme VCKNN

Kode Pengujian	SPR-PT-03
Nomor Skenario	Skenario 3

Tabel 4.26 Kasus Uji Pengujian Performa Skenario 3 Algoritme VCKNN (Lanjutan)

Algoritme	VCKNN	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VCKNN.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.640892,-7.9331800009999]
	Voronoi Tujuan	17
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VCKNN. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

4.7.1.4 Pengujian Performa Skenario 1 Algoritme PINE

Tabel 4.27 merupakan kasus uji pengujian performa pada skenario 1 menggunakan algoritme PINE. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme PINE. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme PINE.

Tabel 4.27 Kasus Uji Pengujian Performa Skenario 1 Algoritme PINE

Kode Pengujian	SPR-PT-04
Nomor Skenario	Skenario 1
Algoritme	PINE
Aktor	Pengguna

Tabel 4.27 Kasus Uji Pengujian Performa Skenario 1 Algoritme PINE (Lanjutan)

Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme PINE.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.60308,-7.946248001]
	Voronoi Tujuan	8
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme PINE. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

4.7.1.5 Pengujian Performa Skenario 2 Algoritme PINE

Tabel 4.28 merupakan kasus uji pengujian performa pada skenario 2 menggunakan algoritme PINE. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme PINE. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme PINE.

Tabel 4.28 Kasus Uji Pengujian Performa Skenario 2 Algoritme PINE

Kode Pengujian	SPR-PT-05
Nomor Skenario	Skenario 2
Algoritme	PINE
Aktor	Pengguna
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme PINE.

Tabel 4.28 Kasus Uji Pengujian Performa Skenario 2 Algoritme PINE (Lanjutan)

Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.623641,-7.949381]
	Voronoi Tujuan	4
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme PINE. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

4.7.1.6 Pengujian Performa Skenario 3 Algoritme PINE

Tabel 4.29 merupakan kasus uji pengujian performa pada skenario 3 menggunakan algoritme PINE. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme PINE.

Tabel 4.29 Kasus Uji Pengujian Performa Skenario 3 Algoritme PINE

Kode Pengujian	SPR-PT-06	
Nomor Skenario	Skenario 3	
Algoritme	PINE	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme PINE.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.640892,-7.9331800009999]
	Voronoi Tujuan	17
	Waktu	Waktu Saat Ini Pada Sistem

Tabel 4.29 Kasus Uji Pengujian Performa Skenario 3 Algoritme PINE (Lanjutan)

Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme PINE. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.

Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme PINE.

4.7.1.7 Pengujian Performa Skenario 1 Algoritme VN3

Tabel 4.30 merupakan kasus uji pengujian performa pada skenario 1 menggunakan algoritme VN3. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VN3. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VN3.

Tabel 4.30 Kasus Uji Pengujian Performa Skenario 1 Algoritme VN3

Kode Pengujian	SPR-PT-07	
Nomor Skenario	Skenario 1	
Algoritme	VN3	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme PINE.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.60308,-7.946248001]
	Voronoi Tujuan	8
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 4.30 Kasus Uji Pengujian Performa Skenario 1 Algoritme VN3 (Lanjutan)

Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VN3. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.

4.7.1.8 Pengujian Performa Skenario 2 Algoritme VN3

Tabel 4.31 merupakan kasus uji pengujian performa pada skenario 2 menggunakan algoritme VN3. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VN3.

Tabel 4.31 Kasus Uji Pengujian Performa Skenario 2 Algoritme VN3

Kode Pengujian	SPR-PT-08	
Nomor Skenario	Skenario 2	
Algoritme	VN3	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VN3.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.623641,-7.949381]
	Voronoi Tujuan	4
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VN3. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VN3.

4.7.1.9 Pengujian Performa Skenario 3 Algoritme VN3

Tabel 4.32 merupakan kasus uji pengujian performa pada skenario 3 menggunakan algoritme VN3. Pengujian dilakukan untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VN3. Hasil yang diharapkan adalah sistem dapat menampilkan *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VN3.

Tabel 4.32 Kasus Uji Pengujian Performa Skenario 3 Algoritme VN3

Kode Pengujian	SPR-PT-09	
Nomor Skenario	Skenario 3	
Algoritme	VN3	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VN3.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.640892,-7.9331800009999]
	Voronoi Tujuan	17
	Waktu	Waktu Saat Ini Pada Sistem
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VN3. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	

4.7.2 Pengujian Statistik *Chi-square*

Pengujian statistik *Chi-square* digunakan untuk melihat perbandingan performa algoritme VCKNN, PINE dan VN3 dengan cara membandingkan nilai *Chi-square* hitung dengan *Chi-square* tabel. Pengujian *Chi-square* dilakukan dengan

membandingkan antara algoritme VCKNN dengan algoritme PINE, algoritme VCKNN dengan algoritme VN3 dan algoritme PINE dengan algoritme VN3.

Pembuatan hipotesis awal dan hipotesis akhir berdasarkan jumlah *node* yang dilalui, dimana jumlah *node* yang lebih kecil dijadikan hipotesis awal sebagai algoritme dengan *response time* lebih baik dari algoritme dengan jumlah *node* yang lebih banyak dan hipotesis akhir bahwa algoritme dengan jumlah *node* lebih banyak memiliki *response time* lebih baik dari algoritme dengan jumlah *node* lebih kecil. Pada pengujian ini menggunakan nilai α=0.1 yang berarti memiliki probabilitas sebesar 90% dan derajat bebas 2 yang didapatkan dari jumlah baris data dikurangi 1, baris data yang digunakan berdasarkan masing-masing algoritme yaitu 3 baris data, sehingga didapatkan *Chi-square* tabel dengan nilai 4.605170186.

Setiap pengujian statistik diberikan kode sebagai identitas, keterangan kodefikasi pengujian statistik sebagai berikut.

Kode Kebutuhan Pengguna: SPR-ST-xx

Keterangan :
SPR : Singkatan dari Sistem Pencarian Rute
ST : Kode representasi pengujian statistik
XX : Nomor urut pengujian statistik

4.7.2.1 Pengujian Statistik Algoritme VCKNN dan PINE

Tabel 4.33 merupakan kasus uji pengujian statistik pada algoritme VCKNN dan PINE. Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme VCKNN dan algoritme PINE.

Tabel 4.33 Kasus Uji Pengujian Statistik Algoritme VCKNN dan PINE

Kode Pengujian	SPR-ST-01	
Algoritme	VCKNN	
	PINE	
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme VCKNN dan algoritme PINE	
Prosedur Uji	1. Penguji mencari nilai *Chi-square* hitung. 2. Penguji membandingkan nilai *Chi-square* hitung dan *Chi-square* tabel.	
Jumlah *Node*	VCKNN	Jumlah *node* yang dilalui menggunakan algoritme VCKNN
	PINE	Jumlah *node* yang dilalui menggunakan algoritme PINE

Tabel 4.33 Kasus Uji Pengujian Statistik Algoritme VCKNN dan PINE (Lanjutan)

Hipotesis	H_0 : Algoritme dengan jumlah *node* yang dilalui terkecil.	
	H_1 : Algoritme dengan jumlah *node* yang dilalui terbesar.	
***Response time* Algoritme VCKNN**	Skenario 1	*Response time* skenario 1 menggunakan algoritme VCKNN
	Skenario 2	*Response time* skenario 2 menggunakan algoritme VCKNN
	Skenario 3	*Response time* skenario 3 menggunakan algoritme VCKNN
***Response time* Algoritme PINE**	Skenario 1	*Response time* skenario 1 menggunakan algoritme PINE
	Skenario 2	*Response time* skenario 2 menggunakan algoritme PINE
	Skenario 3	*Response time* skenario 3 menggunakan algoritme PINE
***Chi-square* Tabel**	4.605170186	
***Chi-square* Hitung**	Hasil dari perhitungan *Chi-square* hitung.	
Hasil	Keputusan yang didapatkan dari membandingkan *Chi-square* hitung dan *Chi-square* tabel, jika *Chi-square* hitung lebih besar dari *Chi-square* tabel maka H_0 ditolak dan H_1 diterima, jika *Chi-square* tabel lebih besar dari *Chi-square* hitung maka H_0 diterima dan H_1 ditolak.	

4.7.2.2 Pengujian Statistik Algoritme VCKNN dan VN3

Tabel 4.34 merupakan kasus uji pengujian statistik pada algoritme VCKNN dan VN3. Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme VCKNN dan algoritme VN3.

Tabel 4.34 Kasus Uji Pengujian Statistik Algoritme VCKNN dan VN3

Kode Pengujian	SPR-ST-02
Algoritme	VCKNN
	VN3
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme VCKNN dan algoritme VN3
Prosedur Uji	1. Penguji mencari nilai *Chi-square* hitung. 2. Penguji membandingkan nilai *Chi-square* hitung dan *Chi-square* tabel.

Tabel 4.34 Kasus Uji Pengujian Statistik Algoritme VCKNN dan VN3 (Lanjutan)

Jumlah *Node*	VCKNN	Jumlah *node* yang dilalui menggunakan algoritme VCKNN
	VN3	Jumlah *node* yang dilalui menggunakan algoritme VN3
Hipotesis	H_0 : Algoritme dengan jumlah *node* yang dilalui terkecil.	
	H_1 : Algoritme dengan jumlah *node* yang dilalui terbesar.	
***Response time* Algoritme VCKNN**	Skenario 1	*Response time* skenario 1 menggunakan algoritme VCKNN
	Skenario 2	*Response time* skenario 2 menggunakan algoritme VCKNN
	Skenario 3	*Response time* skenario 3 menggunakan algoritme VCKNN
***Response time* Algoritme VN3**	Skenario 1	*Response time* skenario 1 menggunakan algoritme VN3
	Skenario 2	*Response time* skenario 2 menggunakan algoritme VN3
	Skenario 3	*Response time* skenario 3 menggunakan algoritme VN3
***Chi-square* Tabel**	4.605170186	
***Chi-square* Hitung**	Hasil dari perhitungan *Chi-square* hitung.	
Hasil	Keputusan yang didapatkan dari membandingkan *chi-squre* hitung dan *Chi-square* tabel, jika *Chi-square* hitung lebih besar dari *Chi-square* tabel maka H_0 ditolak dan H_1 diterima, jika *Chi-square* tabel lebih besar dari *Chi-square* hitung maka H_0 diterima dan H_1 ditolak.	

4.7.2.3 Pengujian Statistik Algoritme PINE dan VN3

Tabel 4.35 merupakan kasus uji pengujian statistik pada algoritme PINE dan VN3. Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme PINE dan algoritme VN3.

Tabel 4.35 Kasus Uji Pengujian Statistik Algoritme PINE dan VN3

Kode Pengujian	SPR-ST-03
Algoritme	PINE
	VN3

Tabel 4.35 Kasus Uji Pengujian Statistik Algoritme PINE dan VN3 (Lanjutan)

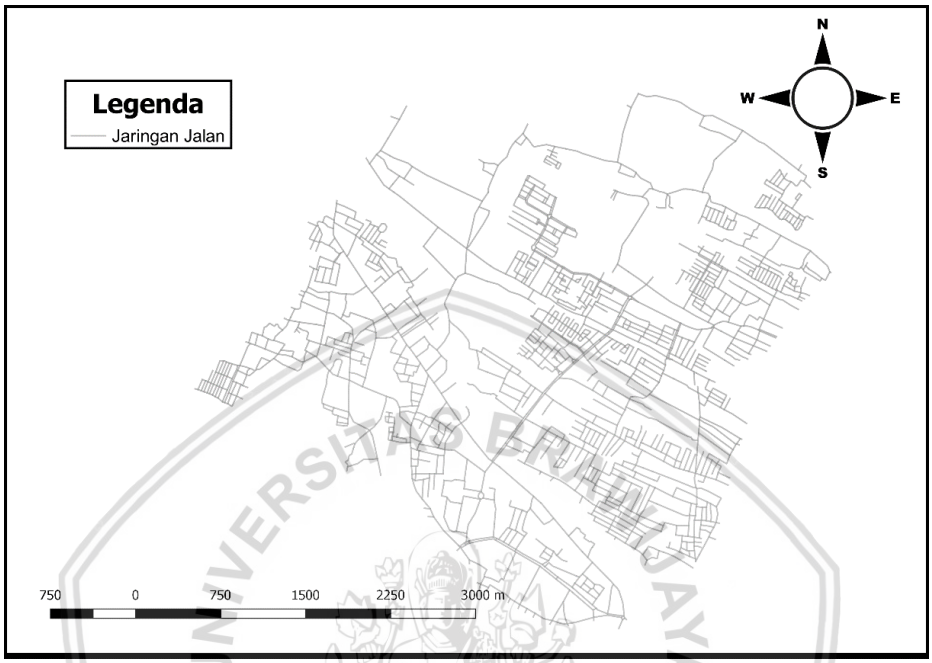
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme PINE dan algoritme VN3	
Prosedur Uji	1. Penguji mencari nilai *Chi-square* hitung. 2. Penguji membandingkan nilai *Chi-square* hitung dan *Chi-square* tabel.	
Jumlah *Node*	PINE	Jumlah *node* yang dilalui menggunakan algoritme PINE
	VN3	Jumlah *node* yang dilalui menggunakan algoritme VN3
Hipotesis	H_0 : Algoritme dengan jumlah *node* yang dilalui terkecil.	
	H_1 : Algoritme dengan jumlah *node* yang dilalui terbesar.	
***Response time* Algoritme PINE**	Skenario 1	*Response time* skenario 1 menggunakan algoritme PINE
	Skenario 2	*Response time* skenario 2 menggunakan algoritme PINE
	Skenario 3	*Response time* skenario 3 menggunakan algoritme PINE
***Response time* Algoritme VN3**	Skenario 1	*Response time* skenario 1 menggunakan algoritme VN3
	Skenario 2	*Response time* skenario 2 menggunakan algoritme VN3
	Skenario 3	*Response time* skenario 3 menggunakan algoritme VN3
***Chi-square* Tabel**	4.605170186	
***Chi-square* Hitung**	Hasil dari perhitungan *Chi-square* hitung.	
Hasil	Keputusan yang didapatkan dari membandingkan *chi-squre* hitung dan *Chi-square* tabel, jika *Chi-square* hitung lebih besar dari *Chi-square* tabel maka H_0 ditolak dan H_1 diterima, jika *Chi-square* tabel lebih besar dari *Chi-square* hitung maka H_0 diterima dan H_1 ditolak.	

4.8 Pengolahan Data

4.8.1 Data Jaringan Jalan dan Waktu Tempuh

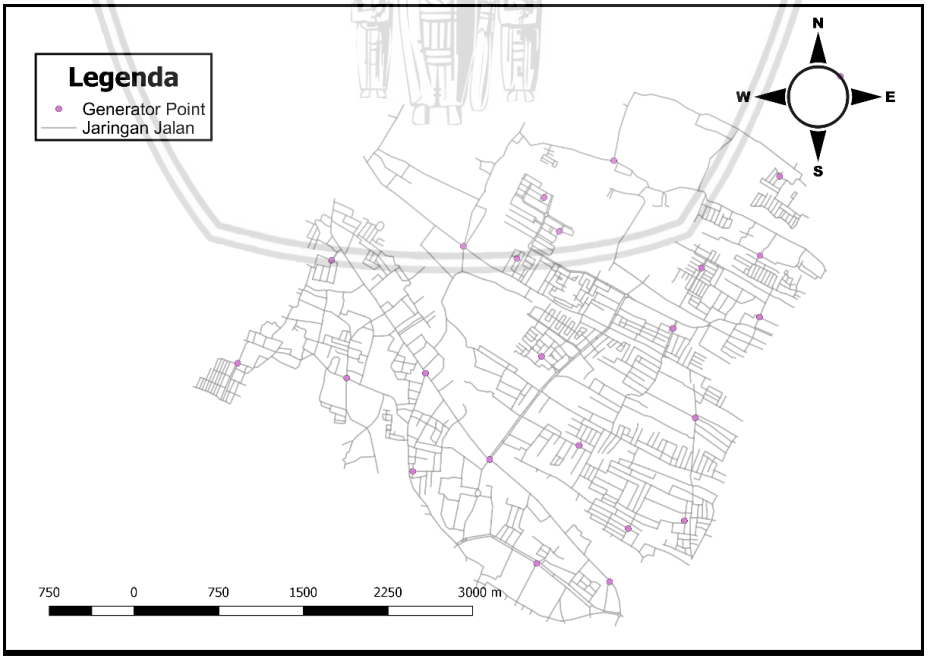
Setelah data diperoleh pada proses pengumpulan data berupa data jaringan jalan, waktu tempuh setiap segmen dan waktu kemacetan, selanjutnya adalah

pengolahan data. Pengolahan data di mulai dari pengolahan data jaringan jalan Kecamatan Lowokwaru. Gambar 4.19 menunjukkan jaringan jalan Kecamatan Lowokwaru, data atribut jarak disetiap segmen diubah menjadi waktu tempuh segmen, nilai tersebut nantinya akan menjadi bobot perhitungan untuk masing-masing metode.



Gambar 4.19 Jaringan Jalan Kecamatan Lowokwaru

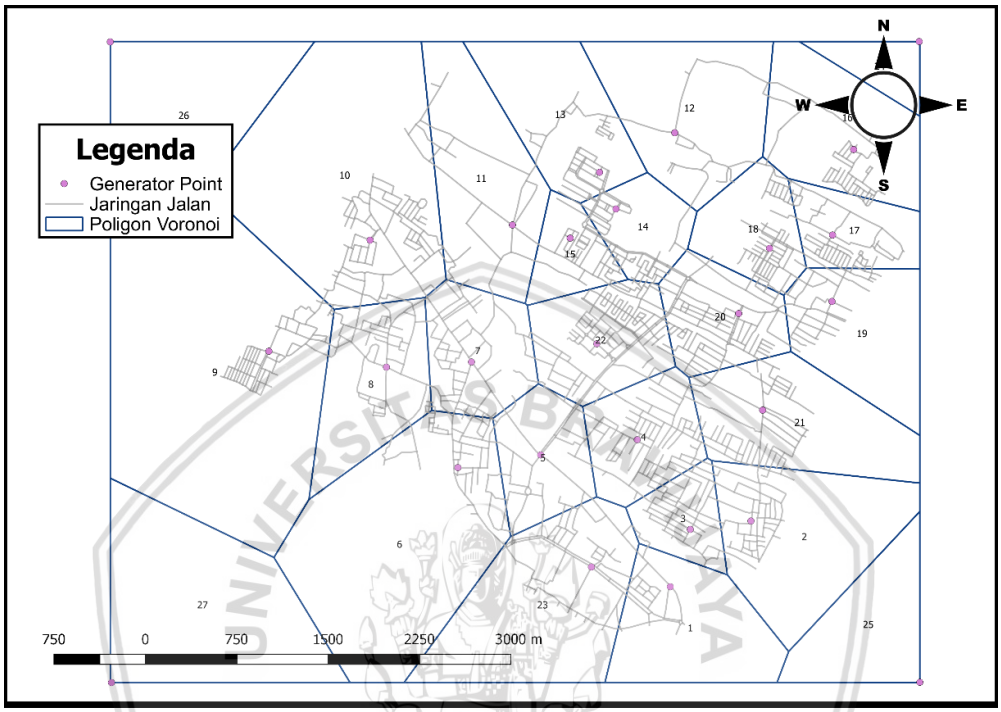
4.8.2 Membuat Titik *Generator Point*



Gambar 4.20 *Generator Point*

Selanjutnya adalah tahapan penentuan *generator point*, *generator point* digunakan sebagai dasar pembuatan poligon *voronoi*. Pada penelitian ini penentuan *generator point* dilakukan secara acak, Gambar 4.20 menunjukan titik-titik yang dijadikan *generator point* pada sistem pencarian rute paling optimum.

4.8.3 Membuat Poligon *Voronoi* dari *Generator Point*

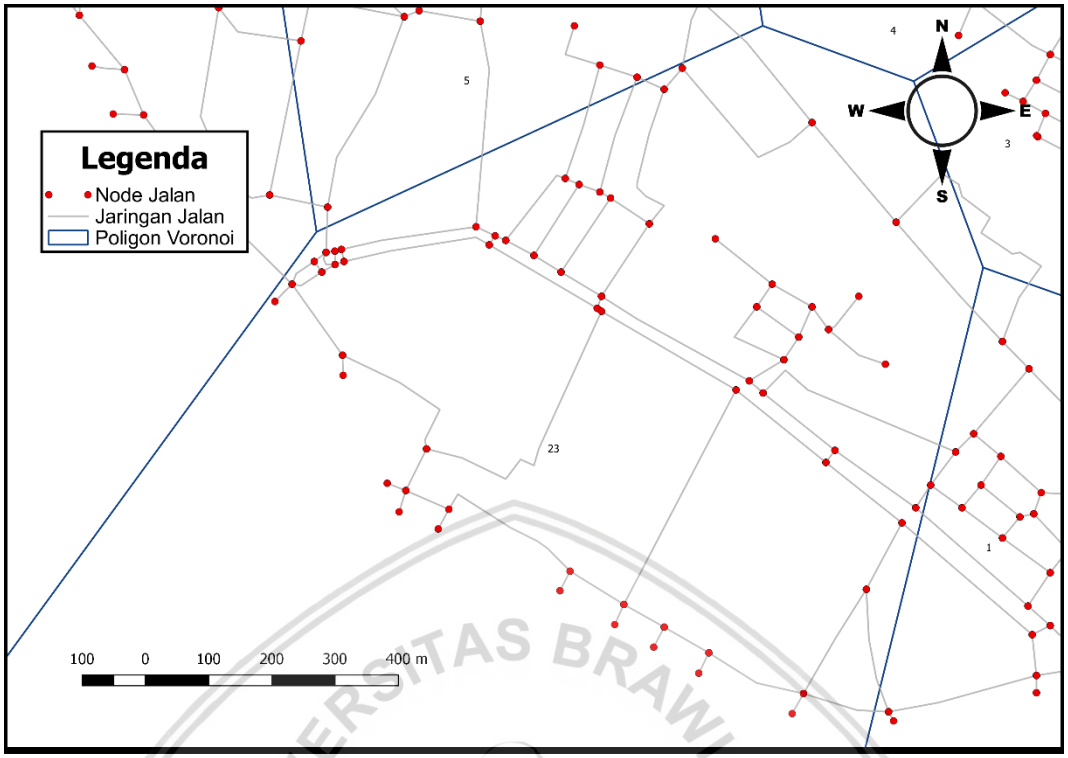


Gambar 4.21 Poligon *Voronoi* Kecamatan Lowokwaru

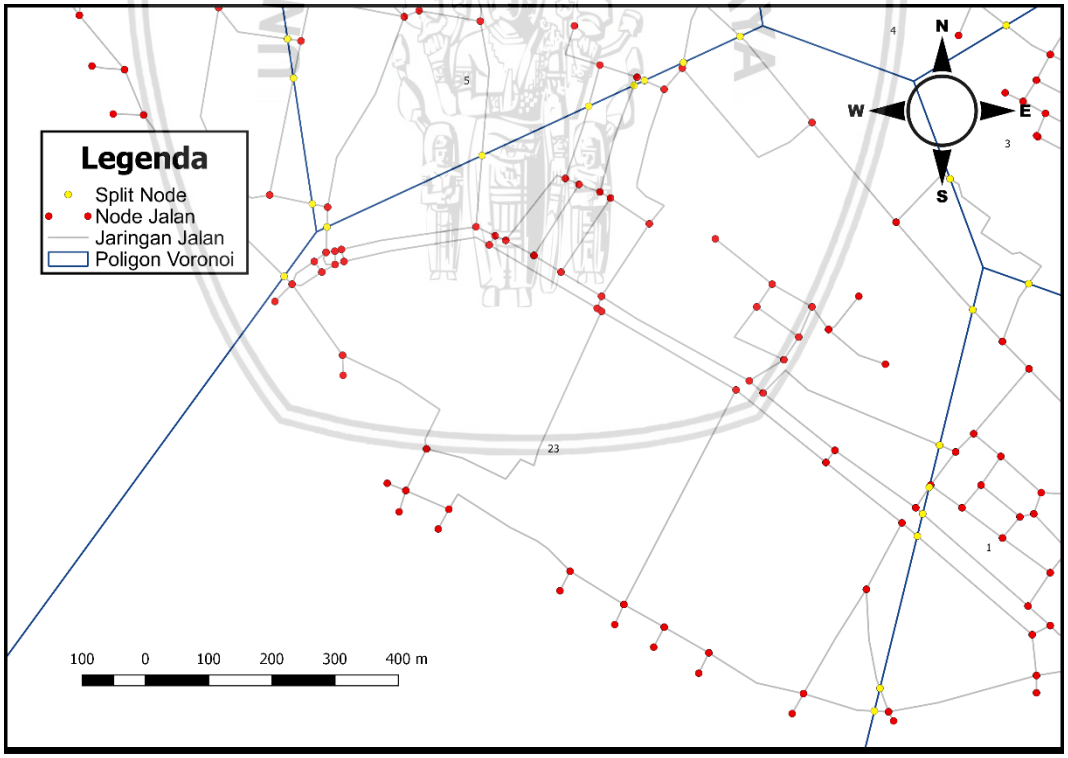
Setelah *generator point* sudah ditentukan, selanjutnya adalah pembuatan poligon *voronoi* berdasarkan *generator point* (Gambar 4.21), pembuatan poligon menggunakan fitur *voronoi polygon* pada *QGIS* dimana *generator point* yang dibuat ditahap sebelumnya dijadikan *input point vector* layer. Fungsi dari poligon *voronoi* adalah untuk mengelompokkan segmen-segmen jalan menjadi segmen yang lebih panjang yang di sebut dengan *Move Interval*.

4.8.4 Memotong Jalan Berdasarkan Poligon *Voronoi*

Setelah poligon *voronoi* terbentuk, selanjutnya adalah proses memotong jalan berdasarkan poligon *voronoi* menggunakan fitur instersect pada QGIS dimana data jaringan jalan sebagai input vector layer dan data poligon *voronoi* sebagai intersect layer. Pada Gambar 4.22 menunjukan data jaringan jalan yang masih belum dipotong, setiap segmen jalan di hubungkan dengan *node-node* yang ditunjukkan dengan titik merah. Gambar 4.23 menunjukkan jaringan jalan yang sudah dipotong, hasil dari pemotongan ini adalah munculnya *split node*. *Split node* adalah *node-node* pada jalan yang berpotongan dengan poligon *voronoi* yang ditunjukkan dengan titik kuning (Gambar 4.23). *Split node* akan digunakan untuk segmen-segmen jalan baru yang di sebut *move interval*.



Gambar 4.22 Segmen Jalan



Gambar 4.23 Segmen Jalan Dengan *Split Node*

4.8.5 *Export* Jaringan Jalan Menjadi *WebGIS*

Setelah melakukan pemotongan jalan berdasarkan poligon *voronoi*, data jaringan jalan di export menjadi *WebGIS* menggunakan fitur qgis2web, hasil dari

tahapan ini adalah data jaringan jalan dalam bentuk file geojson yang akan di tampilkan ke *WebGIS*.

4.8.6 *Import* Data Jaringan Jalan Dalam Bentuk Kedalam *Database*

Setelah *export* data jaringan jalan menjadi *WebGIS*, data jaringan jalan masih berupa file *geojson*, tahapan selanjutnya adalah menyimpan data jaringan jalan kedalam *database*, tujuannya adalah untuk memudahkan pengolahan data pada pencarian rute paling optimum serta pencarian tetangga dan pencarian *move interval*. Hasil dari tahapan ini adalah data jaringan jalan yang tersimpan pada *database*.

4.8.7 Mencari Tetangga *Voronoi* Dan Menyimpannya Ke Dalam *Database*

Setelah data jaringan jalan tersimpan pada *database*, tahapan selanjutnya adalah mencari tetangga *voronoi* serta meyimpannya pada *database*. Tujuan dari tahapan ini adalah untuk mencari tetangga dari poligon *voronoi* yang nantinya digunakan untuk algoritme VCKNN, VN3 dan PINE dalam pencarian rute paling optimum. Hasil dari tahapan ini adalah data *voronoi* beserta tetangganya yang disimpan pada *database*.

4.8.8 Mencari *Move Interval* Dan Menyimpannya Ke *Database*

Tujuan dari tahapan ini adalah mencari *move interval* pada masing-masing poligon *voronoi*. *Move interval* adalah segmen yang menghubungkan setiap *split node*, penentuan jalur untuk *move interval* menggunakan algoritme dijkstra. Hasil dari tahapan ini adalah data *move interval* yang tersimpan dalam *database*.

BAB 5 IMPLEMENTASI

Bab implementasi menjelaskan tentang kebutuhan implementasi sistem pencarian rute paling optimum dan hasil implementasi sistem pencarian rute paling optimum berdasarkan dari hasil perancangan sistem pada bab perancangan. Pada bab implentasi berisi spesifikasi lingkungan implementasi, implementasi database, implementasi algoritme dan implementasi antarmuka.

5.1 Spesifikasi Lingkungan Implementasi

Subbab spesifikasi lingkungan implementasi menjelaskan spesifikasi dari perangkat keras serta perangkat lunak yang digunakan dalam proses implementasi sistem pencarian rute paling optimum. Dalam proses pembangunan sistem, Perangkat keras yang digunakan adalah laptop asus n46vj yang dijelaskan pada Tabel 5.1, untuk perangkat lunak yang digunakan sebagai *server* adalah apache yang dijelaskan pada Tabel 5.2 dan untuk lingkungan *deployment* dijelaskan pada Tabel 5.3.

Tabel 5.1 Spesifikasi Perangkat Keras

Unit Komputasi	Laptop ASUS N46VJ
CPU	Core i5-3210M 2.5 Ghz
Kapasitas RAM	8 Gb
Kapasitas Penyimpanan	500 Gb
Kartu Grafis	NVidia GeForce 635M
Resolusi Layar	1366 x 768 pixels

Tabel 5.2 Spesifikasi Perangkat Lunak

Sistem Operasi	Windows 7 Ultimate 64-bit
Web Server	Apache 2.4.12
DBMS	MySQL 5.6.24
Bahasa Pemrograman	PHP 5.5.24
Editor Kode Program	Notepad++
Peramban	Google Chrome Version 67.0.3396.99 (Official Build) (64-bit)
Perangkat Lunak atau Bahasa Pemrogarman Pendukung	Javascript
	HTML

Tabel 5.3 Spesifikasi Minimal Lingkungan *Deployment*

Kapasitas RAM (Minimal)	256 MB
Kapasitas Memori (Minimal)	85 MB
Versi PHP	5.3 atau lebih

5.2 Implementasi Algoritme

Dalam sistem pencarian rute paling optimum terdapat 3 algoritme yang digunakan, yaitu VCKNN, PINE dan VN3. Berdasarkan perancangan diagram kelas terdapat 5 kelas dalam sistem pencarian rute paling optimum, yaitu kelas dijkstra yang digunakan untuk mencari jalur, kelas voronoi yang digunakan untuk mencari tetangga *voronoi*, kelas vcknn yang digunakan untuk mencari rute mengunakan algoritme vcknn, kelas pine yang digunakan untuk mencari rute menggunakan algoritme pine dan kelas vn3 yang digunakan untuk mencari rute menggunakan algoritme vn3. Untuk menjalankan algoritme VCKNN, PINE dan VN3 dibutuhkan kelas dijkstra untuk pencarian jalur dan kelas voronoi untuk mencari tetangga *voronoi*.

5.2.1 Implementasi Kelas Dijkstra

Pada kelas dijkstra terdapat fungsi-fungsi untuk pencarian jalur, kelas dijkstra merupakan kelas yang digunakan untuk mencari jalur dengan bobot waktu tempuh setiap segmen jalan pada algoritme VCKNN, PINE dan VN3. Tabel 5.4 merupakan kode program dari kelas dijkstra.

Tabel 5.4 Kode Program Kelas Dijkstra

No	Kode Program
1	`class Dijkstra extends CI_Controller {`
2	`private $dataJalan=array();`
3	`private $B=array();`
4	`private $J=array();`
5	`private $S=array();`
6	`private $idWaktu=0;`
7	`private $waktu_m='s';`
8	`private $jalan_m='s';`
9	`private $mi_m='s';`
10	`function __construct($ins=''){`
11	`parent::__construct();`
12	`$this->waktu_m='waktu'.$ins;`
13	`$this->jalan_m='jalan'.$ins;`
14	`$this->mi_m='mi'.$ins;`
15	`$this->load->model('M_waktu',$this->waktu_m);`
16	`$this->load->model('M_jalanv',$this->jalan_m);`
17	`$this->load->model('M_mi',$this->mi_m);`
18	`}`
19	`function resetData(){`
20	`$this->dataJalan=array();`
21	`$this->B=array();`
22	`$this->J=array();`
23	`$this->S=array();`
24	`$this->idWaktu =0;`

Tabel 5.4 Kode Program Kelas Dijkstra (Lanjutan)

```
25          }
26          function setIdWaktu($hari='',$jam=0){
27                  $this->idWaktu=$this->{$this->waktu_m}-
    >getWaktu($hari,$jam);
28          }
29          function setDataJalanV($idv){
30                  $getJalan =$this->{$this->jalan_m}-
    >getJalan($idv,$this->idWaktu);
31                  $this->fecthData($getJalan);
32          }
33          function setDataJalanMI($idv){
34                  $getMoveInterval =$this->{$this->mi_m}-
    >getMi($idv,$this->idWaktu);
35                  $this->fecthData($getMoveInterval);
36          }
37          function fecthData($getJalan){
38                  foreach ($getJalan as $barisJalan) {
39                          $this-
    >dataJalan[$barisJalan['simpul_awal']][]=array($barisJalan['simpu
    l_akhir'],$barisJalan['waktu'],$barisJalan['id_jalan']);
40                          $this-
    >dataJalan[$barisJalan['simpul_akhir']][]=array($barisJalan['simp
    ul_awal'],$barisJalan['waktu'],$barisJalan['id_jalan']);
41                          if(in_array($barisJalan['simpul_awal'],$this-
    >B)==false){
42                                  $this->B[]=$barisJalan['simpul_awal'];
43                                  $this-
    >J[$barisJalan['simpul_awal']]=array(0,'T',0);
44                          }
45                         if(in_array($barisJalan['simpul_akhir'],$this-
    >B)==false){
46                                  $this-
    >B[]=$barisJalan['simpul_akhir'];
47                                  $this-
    >J[$barisJalan['simpul_akhir']]=array(0,'T',0);
48                          }
49                  }
50          }
51          function fecthDataArray($getJalan){
52                  foreach($getJalan as $ML){
53                          foreach($ML as $barisJalan){
54                                  $this-
    >dataJalan[$barisJalan['awal']][]=array($barisJalan['akhir'],$bar
    isJalan['waktu'],$barisJalan['id_jalan']);
55                                  $this-
    >dataJalan[$barisJalan['akhir']][]=array($barisJalan['awal'],$bar
    isJalan['waktu'],$barisJalan['id_jalan']);
56                                  if(in_array($barisJalan['awal'],$this-
    >B)==false){
57                                          $this->B[]=$barisJalan['awal'];
58                                          $this-
    >J[$barisJalan['awal']]=array(0,'T',0);
59                                  }
60                                 if(in_array($barisJalan['akhir'],$this-
    >B)==false){
61                                          $this->B[]=$barisJalan['akhir'];
62                                          $this-
    >J[$barisJalan['akhir']]=array(0,'T',0);
63                                  }
64                          }
65                  }
66          }
```

Tabel 5.4 Kode Program Kelas Dijkstra (Lanjutan)

```
67          function cariJalur($nodeAwal,$nodeTujuan,$idv)){
68                  $C=array($nodeAwal,0,0);
69                  $MC=array();
70                  $this->J[$C[0]]=array(0,0,0);
71                  while(count($this->B)!=0){
72                          $MC=array(0,0);
73                          $this->S[]=$C[0];
74                          $indeks = array_search($C[0], $this->B);
75                          unset($this->B[$indeks]);
76                          for($i=0;$i<count($this-
    >dataJalan[$C[0]]);$i++){
77                                  if(in_array($this-
    >dataJalan[$C[0]][$i][0],$this->B)==true){
78                                          $tempWaktu=$this-
    >dataJalan[$C[0]][$i][1]+$C[1];
79                                          if($this->J[$this-
    >dataJalan[$C[0]][$i][0]][1] === 'T'){
80                                                  $this->J[$this-
    >dataJalan[$C[0]][$i][0]]=array($C[0],$tempWaktu,$this-
    >dataJalan[$C[0]][$i][2]);
81                                          }
82                                          else if ($tempWaktu<$this-
    >J[$this->dataJalan[$C[0]][$i][0]][1]){
83                                                  $this->J[$this-
    >dataJalan[$C[0]][$i][0]]=array($C[0],$tempWaktu,$this-
    >dataJalan[$C[0]][$i][2]);
84                                          }
85                                  }
86                          }
87                          foreach ($this->B as $tempB) {
88                                  if($this->J[$tempB][1] !== 'T'){
89                                          if($MC[0]===0){
90                                                  $MC=array($tempB,$this-
    >J[$tempB][1]);
91                                          }
92                                          else if($this-
    >J[$tempB][1]<$MC[1]){
93                                                  $MC=array($tempB,$this-
    >J[$tempB][1]);
94                                          }
95                                  }
96                          }
97                          if($MC[0]===0){
98                                  break;
99                          }
100                         $C=$MC;
101                 }
102                 if(is_array($nodeTujuan)){
103                         $hasilnya=array();
104                         foreach ($nodeTujuan as $tempNode){
105                                 $akhir=$tempNode;
106                                 $tempID=$this->J[$tempNode][2];
107                                 if($this->J[$akhir][0]!==0 ){
108                                         $waktu=$this->J[$akhir][1];
109                                         if($akhir != $nodeAwal){
110                                                 $node=$akhir;
111                                                 while($akhir!=$nodeAwal){
112                                                         $akhir=$this-
    >J[$akhir][0];
113                                                 $node=$akhir.','.$node;
114                                                 $tempID=$tempID.','.$this-
    >J[$akhir][2];
115                                                 }
```

Tabel 5.4 Kode Program Kelas Dijkstra (Lanjutan)

```
116  $hasilnya[]=array('awal'=>$nodeAwal,'akhir'=>$tempNode,'node'=>$n
     ode,'id_jalan'=>substr($tempID, 0, -
     2),'waktu'=>$waktu,'idv'=>$idv[0]);
117                                        }
118                                  }
119                            }
120                            return $hasilnya;
121                      }
122              }
123      }
```

5.2.2 Implementasi Kelas Voronoi

Pada kelas voronoi terdapat fungsi-fungsi untuk pencarian tetangga *voronoi*, kelas voronoi merupakan kelas yang digunakan untuk mencari mencari tetangga *voronoi* pada algoritme VCKNN, PINE dan VN3. Tabel 5.5 merupakan kode program dari kelas voronoi.

Tabel 5.5 Kode Program Kelas Voronoi

No	Kode Program
1	include "Dijkstra.php";
2	class Voronoi extends CI_Controller{
3	function __construct(){
4	parent::__construct();
5	$this->load->model('M_voronoi');
6	$this->load->model('M_jalanv');
7	$this->load->model('M_mi');
8	}
9	function getSimpulUjung($idv=''){
10	$getVoronoi = $this->M_jalanv->getIdv($idv);
11	$data=array();
12	foreach ($getVoronoi as $dataVoronoi) {
13	$idv=$dataVoronoi['idv'];
14	$getTitik = $this->M_jalanv->getTitik($idv);
15	foreach ($getTitik as $dataTitik) {
16	$data[$idv][]=$dataTitik['simpul'];
17	}
18	}
19	return $data;
20	}
21	function cariVoronoi($vAwal,$vTujuan){
22	$tempT=array();
23	$tempS=array();
24	$tempB=array();
25	$getVoronoi = $this->M_voronoi->getVoronoi();
26	$dataVoronoi=array();
27	foreach ($getVoronoi as $barisVoronoi) {
28	$dataVoronoi[$barisVoronoi['voronoi']][]=$barisVoronoi['te tangga'];
29	}
30	array_push($tempB,$vAwal);
31	$selesai=false;
32	while(count($tempB)!=0){
33	for($i=0;$i<count($dataVoronoi[$tempB[0]]);$i++){
34	if(in_array($tempB[0],$tempS)==false){
35	array_push($tempB,$dataVoronoi[$tempB[0]][$i]);
36	if($dataVoronoi[$tempB[0]][$i]!=$vAwal){
37	if($dataVoronoi[$tempB[0]][$i]!=$vTujuan){
38	if(in_array($dataVoronoi[$tempB[0]][$i],$tempT)==false){
39	array_push($tempT,$dataVoronoi[$tempB[0]][$i]);

Tabel 5.5 Kode Program Kelas Voronoi (Lanjutan)

```
40                                                          }
41                                                  }
42                                          }
43                          if ($dataVoronoi[$tempB[0]][$i]==$vTujuan){
44                                                  $selesai=true;
45                                          }
46                                  }
47                                  else {
48                                          break;
49                                  }
50                          }
51                          if($selesai){break;}
52                          array_push($tempS,$tempB[0]);
53                          array_shift($tempB);
54                  }
55                  return $tempT;
56          }
57          function createMI($vAwal,$vTujuan,$idWaktu){
58                  $splitNAwal=$this->getSimpulUjung(" where
    idv=".$vAwal[1]." ")[$vAwal[1]];
59                  $splitNAkhir=$this->getSimpulUjung(" where
    idv=".$vTujuan[1]." ")[$vTujuan[1]];
60                  $ml=array();
61                  $d1 = new Dijkstra('1');
62                  $d1->resetData();
63                  $d1->setIdWaktu($idWaktu[0],$idWaktu[1]);
64                  $d1->setDataJalanV(array($vAwal[1]));
65                  $ml[]=$d1-
    >cariJalur($vAwal[0],$splitNAwal,array($vAwal[1]));
66                  foreach($splitNAkhir as $node){
67                          $d1->resetData();
68                          $d1->setIdWaktu($idWaktu[0],$idWaktu[1]);
69                          $d1->setDataJalanV(array($vTujuan[1]));
70                          $tempArray=$d1-
    >cariJalur($node,array($vTujuan[0]),array($vTujuan[1]));
71                          if (empty($tempArray)==false) {
72                                  $ml[]=$tempArray;
73                          }
74                  }
75                  return $ml;
76          }
77  }
```

5.2.3 Implementasi Algoritme VCKNN

Pada algoritme VCKNN dibagi menjadi 2 tahap yaitu pencarian tetangga *voronoi* dan pencarian jalur dengan bobot terkecil, pencarian tetangga *voronoi* menggunakan fungsi-fungsi pada kelas voronoi dan pencarian jalur menggunakan fungsi-fungsi pada kelas dijkstra. Dalam algoritme VCKNN menggunakan data segmen jalan *voronoi* pada *voronoi* awal dan *voronoi* tujuan serta menggunakan data *move interval* pada *voronoi* tetangga. Algoritme VCKNN dimulai dari memasukkan *node* awal, *voronoi* awal, *node* tujuan dan *voronoi* tujuan, kemudian mencari *voronoi* tetangga dan inisialisasi data *move interval* pada *voronoi* tetangga, lalu inisialisasi data jalan *voronoi* pada *voronoi* awal dan *voronoi* tujuan, setelah itu melakukan pencarian jalur. Tabel 5.6 merupakan implementasi dari algoritme VCKNN.

Tabel 5.6 Implementasi Algoritme VCKNN

No	Kode Program
1	`function main(){`
2	`$lokasi=$this->input->post('lokasi');`
3	`$data=json_decode($lokasi,1);`
4	`$nodeAwal=substr(substr($data[0][0], 0, -1), 1);`
5	`$vAwal=$data[0][1];`
6	`$nodeTujuan=substr(substr($data[1][0], 0, -1), 1);`
7	`$vTujuan=$data[1][1];`
8	`$d = new Dijkstra();`
9	`$d->resetData();`
10	`$d->setIdWaktu($data[2],$data[3]);`
11	`$v= new Voronoi();`
12	`if($vAwal!=$vTujuan){`
13	`$tempT=$v->cariVoronoi($vAwal,$vTujuan);`
14	`$d->setDataJalanMI($tempT);`
15	`}`
16	`$d->setDataJalanV(array($vAwal,$vTujuan));`
17	`$jalan=$d->cariJalur($nodeAwal,array($nodeTujuan));`
18	`echo json_encode($jalan[0]);`
19	`}`

5.2.4 Implementasi Algoritme PINE

Pada algoritme PINE dibagi menjadi 2 tahap yaitu pencarian tetangga *voronoi* dan pencarian jalur dengan bobot terkecil, pencarian tetangga *voronoi* menggunakan fungsi-fungsi pada kelas voronoi dan pencarian jalur menggunakan fungsi-fungsi pada kelas dijkstra. Dalam algoritme PINE menggunakan data *move interval* pada *voronoi* awal, *voronoi* tujuan dan *voronoi* tetangga.

Tabel 5.7 Implementasi Algoritme PINE

No	Kode Program
1	`function main(){`
2	`$lokasi=$this->input->post('lokasi');`
3	`$data=json_decode($lokasi,1);`
4	`$nodeAwal=substr(substr($data[0][0], 0, -1), 1);`
5	`$vAwal=$data[0][1];`
6	`$nodeTujuan=substr(substr($data[1][0], 0, -1), 1);`
7	`$vTujuan=$data[1][1];`
8	`$d = new Dijkstra();`
9	`$d->resetData();`
10	`$d->setIdWaktu($data[2],$data[3]);`
11	`$v= new Voronoi();`
12	`if($vAwal!=$vTujuan){`
13	`$tempT=$v->cariVoronoi($vAwal,$vTujuan);`
14	`$d->setDataJalanMI($tempT);`
15	`}`
16	`$ML=$v->createMI(array($nodeAwal,$vAwal),array($nodeTujuan,$vTujuan),array($data[2],$data[3]));`
17	`$d->fecthDataArray($ML);`
18	`$jalan=$d->cariJalur($nodeAwal,array($nodeTujuan));`
19	`echo json_encode($jalan[0]);`
20	`}`

Algortima PINE dimulai dari memasukkan *node* awal, *voronoi* awal, *node* tujuan dan *voronoi* tujuan, kemudian *mencari* voronoi tetangga, lalu inisialisasi data *move interval* pada *voronoi* awal, *voronoi* tujuan dan *voronoi* tetangga, setelah itu

melakukan pencarian jalur. Tabel 5.7 merupakan implementasi dari algoritme PINE.

5.2.5 Implementasi Algoritme VN3

Pada algoritme VN3 dibagi menjadi 2 tahap yaitu pencarian tetangga *voronoi* dan pencarian jalur dengan bobot terkecil, pencarian tetangga *voronoi* menggunakan fungsi-fungsi pada kelas voronoi dan pencarian jalur menggunakan fungsi-fungsi pada kelas dijkstra. Dalam algoritme VN3 menggunakan data segmen jalan *voronoi* pada *voronoi* awal, *voronoi* tujuan dan *voronoi* tentangga. Algoritme VN3 dimulai dari memasukkan *node* awal, *voronoi* awal, *node* tujuan dan *voronoi* tujuan, kemudian mencari *voronoi* tetangga, lalu inisialisasi data jalan *voronoi* pada *voronoi* awal, *voronoi* tujuan dan *voronoi* tetangga, setelah itu melakukan pencarian jalur. Tabel 5.8 merupakan implementasi dari algoritme VN3.

Tabel 5.8 Implementasi Algoritme VN3

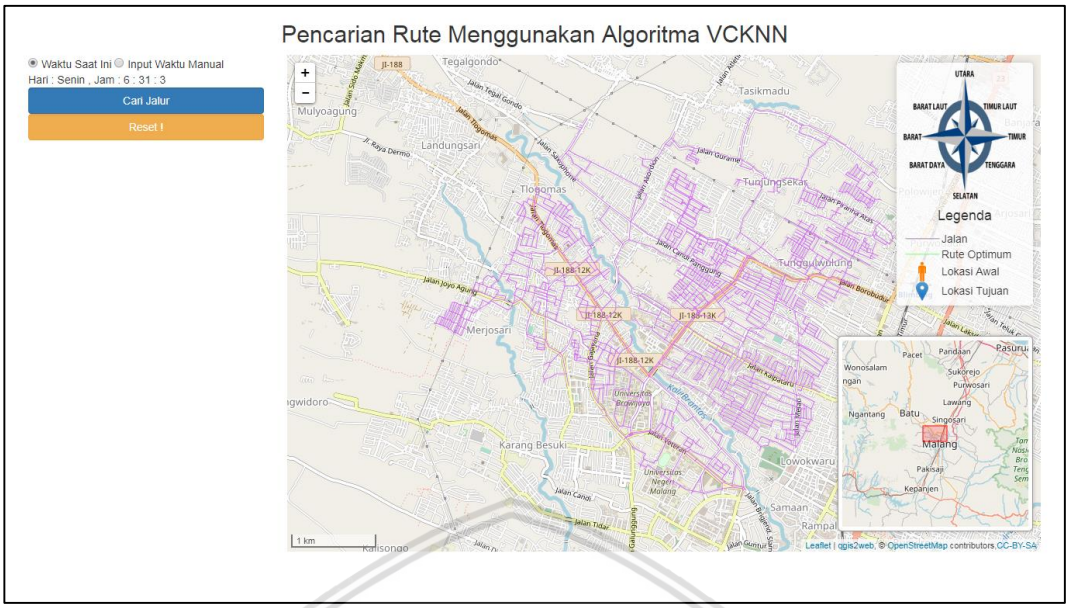
No	Kode Program
1	`function main(){`
2	`$lokasi=$this->input->post('lokasi');`
3	`$data=json_decode($lokasi,1);`
4	`$nodeAwal=substr(substr($data[0][0], 0, -1), 1);`
5	`$vAwal=$data[0][1];`
6	`$nodeTujuan=substr(substr($data[1][0], 0, -1), 1);`
7	`$vTujuan=$data[1][1];`
8	`$d = new Dijkstra();`
9	`$d->resetData();`
10	`$d->setIdWaktu($data[2],$data[3]);`
11	`$v= new Voronoi();`
12	`$tempT=array();`
13	`if($vAwal!=$vTujuan){`
14	`$tempT=$v->cariVoronoi($vAwal,$vTujuan);`
15	`}`
16	`array_push($tempT,$vAwal,$vTujuan);`
17	`$d->setDataJalanV($tempT);`
18	`$jalan=$d->cariJalur($nodeAwal,array($nodeTujuan));`
19	`echo json_encode($jalan[0]);`
20	`}`

5.3 Implementasi Antarmuka

Pada implementasi antarmuka menjelaskan hasil dari implementasi antarmuka pengguna berdasarkan perancangan antarmuka sistem pencarian rute paling optimum berupa potongan gambar. Implementasi antarmuka pengguna terdiri dari antarmuka menampilkan peta dan antarmuka pencarian rute.

5.3.1 Antarmuka Menampilkan Peta

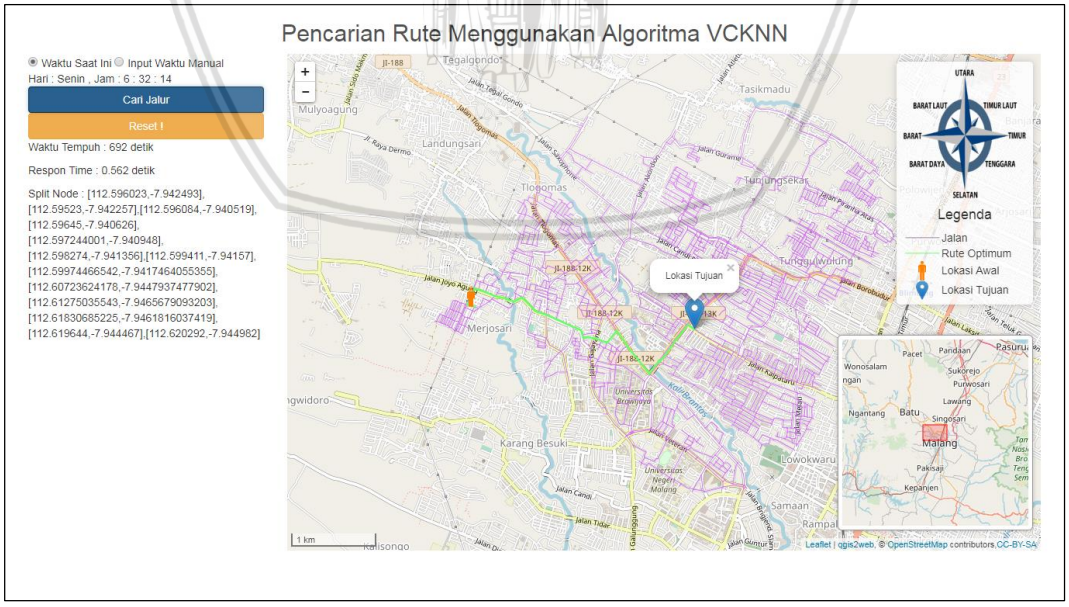
Gambar 5.1 merupakan antarmuka menampilkan peta yang ditampilkan pertama kali saat pengguna menggunakan sistem pencarian rute paling optimum, antarmuka menampilkan peta terdiri dari judul algoritme yang digunakan dalam pencarian jalur, navigasi untuk memasukan data yang digunakan dalam pencarian rute, dan *WebGIS* untuk memberikan informasi jaringan jalan Kecamatan Lowokwaru.



Gambar 5.1 Antarmuka Menampilkan Peta

5.3.2 Antarmuka Pencarian Rute

Gambar 5.2 merupakan antarmuka pencarian rute yang ditampilkan saat pengguna memasukan lokasi awal, lokasi tujuan dan mendefinisikan waktu serta menekan tombol cari jalur. Antarmuka pencarian rute terdiri dari judul algoritme yang digunakan dalam pencarian jalur, navigasi untuk memasukan data yang digunakan dalam pencarian rute, keterangan hasil pencarian jalur serta *WebGIS* untuk memberikan informasi jaringan jalan Kecamatan Lowokwaru dan rute paling optimum yang dirokemendasikan sistem.



Gambar 5.2 Antarmuka Mencari Rute

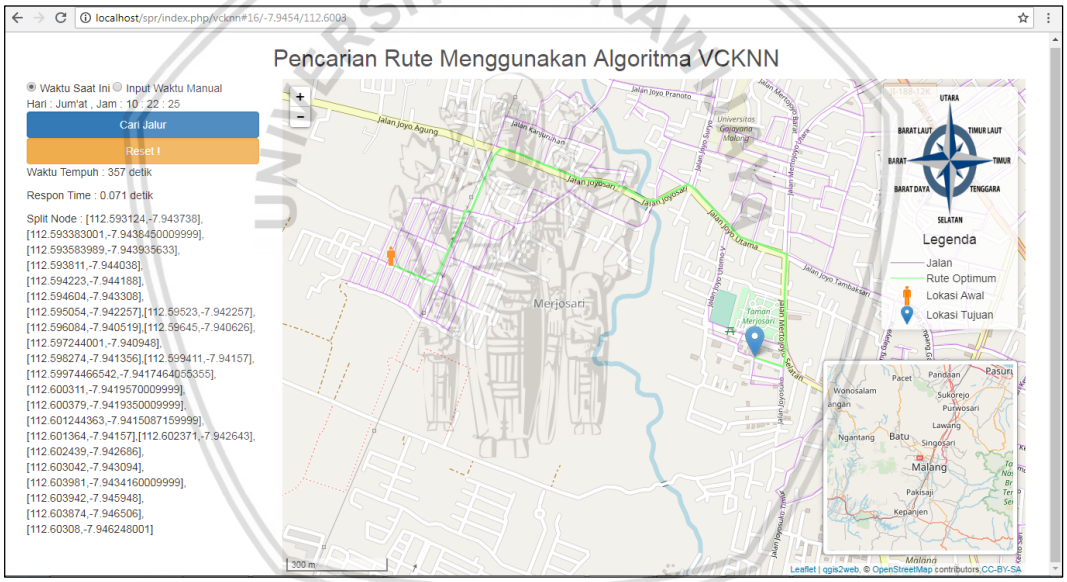
BAB 6 PENGUJIAN

Bab pengujian menjelaskan hasil dari pengujian dan evaluasi hasil pengujian pada sistem pencarian rute paling optimum berdasarkan perancangan pengujian. Pengujian dalam sistem pencarian rute paling optimum terdiri dari pengujian performa pada aspek *response time* untuk mengetahui waktu yang dibutuhkan sistem untuk menganggapi permintaan pengguna pada masing-masing algoritme dan pengujian statistik *Chi-square* untuk mengetahui perbandingan performa masing-masing algoritme.

6.1 *Performance Testing*

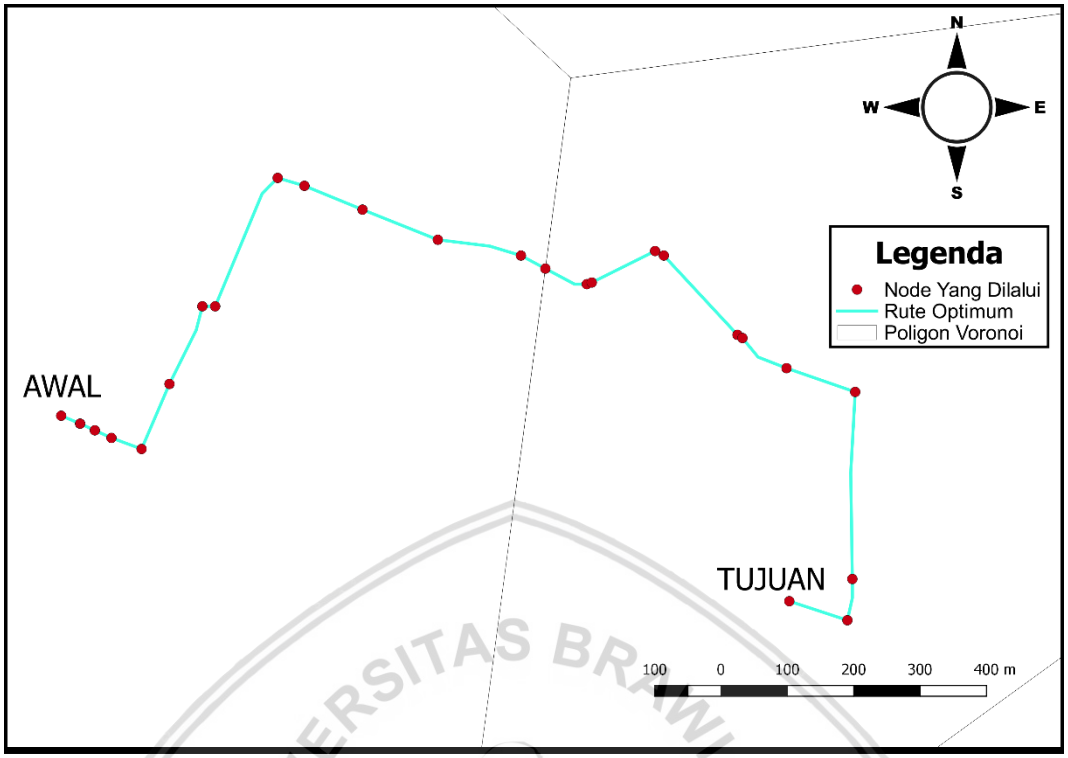
Pengujian performa dilakukan berdasarkan perancangan pengujian performa, pada pengujian performa dilakukan 9 pengujian performa, pengujian dilakukan menggunakan 3 skenario pada masing-masing algoritme.

6.1.1 Pengujian Performa Skenario 1 Algoritme VCKNN



Gambar 6.1 Skenario 1 Algoritme VCKNN

Gambar 6.1 merupakan hasil pengujian skenario 1 pada algoritme VCKNN. Dalam algoritme VCKNN dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.60308,-7.946248001]" memiliki *response time* 0.071 detik dan waktu tempuh 357 detik. Pada skenario 1 menggunakan algoritme VCKNN terdapat 25 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.1), Gambar 6.2 menjelaskan *node-node* yang dilalui pada skenario 1 menggunakan algoritme VCKNN. Tabel 6.1 menjelaskan hasil pengujian performa skenario 1 menggunakan algoritme VCKNN.



Gambar 6.2 Skenario 1 Algoritme VCKNN

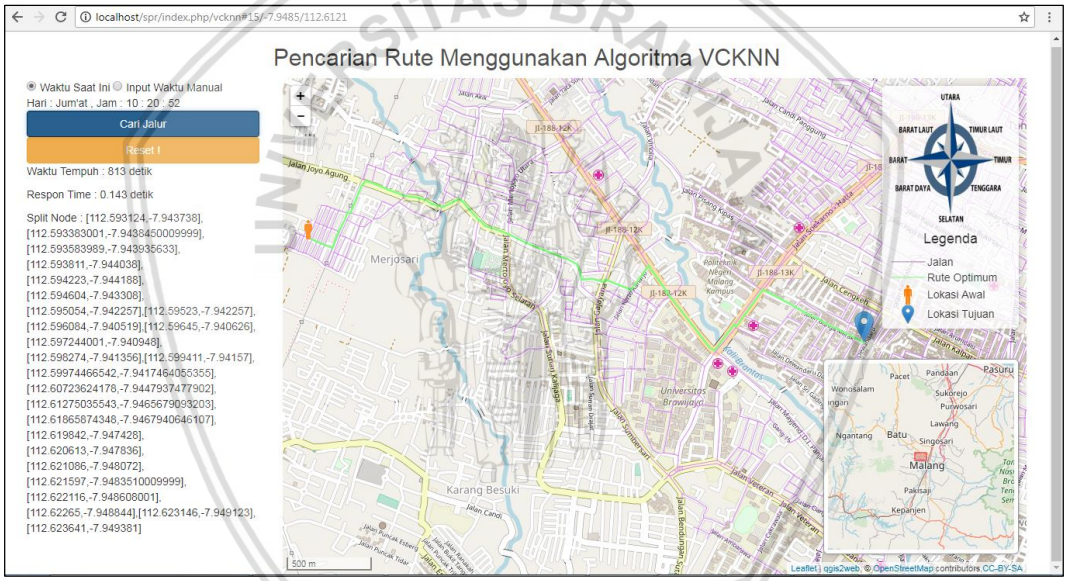
Tabel 6.1 Hasil Pengujian Performa Skenario 1 Algoritme VCKNN

Kode Pengujian	SPR-PT-01	
Nomor Skenario	Skenario 1	
Algoritme	VCKNN	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VCKNN.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.60308,-7.946248001]
	Voronoi Tujuan	8
	Waktu	Hari : Jum'at , Jam : 10 : 22 : 50
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.1 Hasil Pengujian Performa Skenario 1 Algoritme VCKNN (Lanjutan)

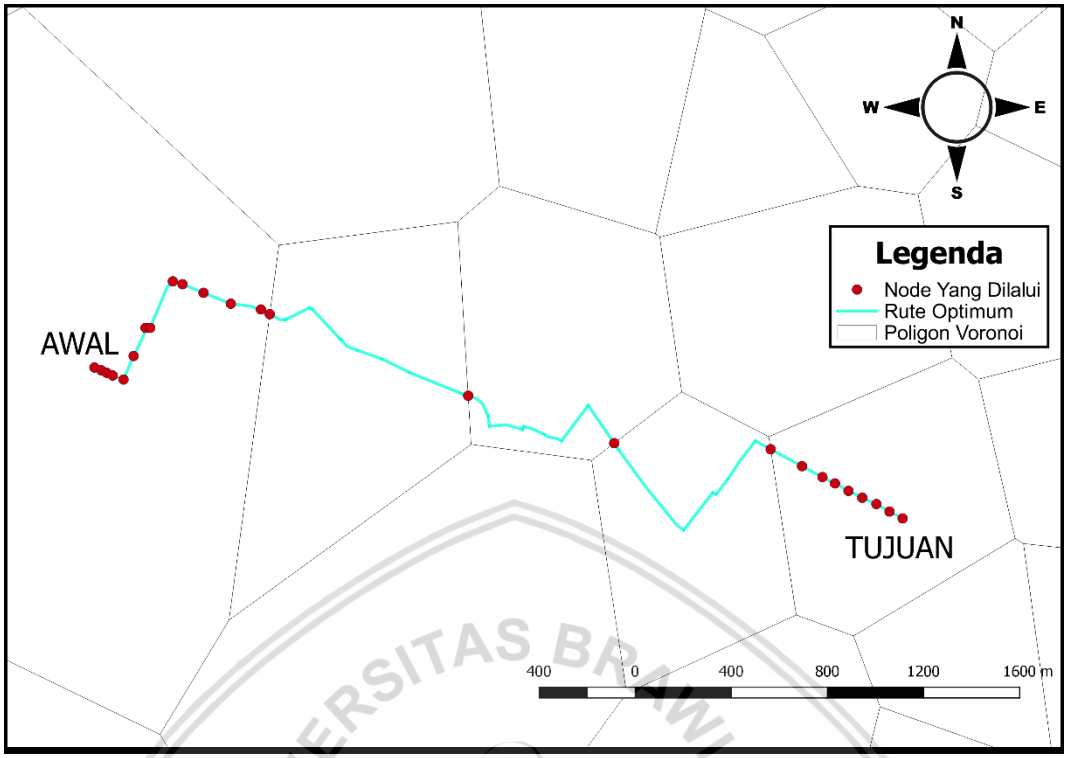
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VCKNN. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.071 detik
	Waktu Tempuh	357 detik
	Jumlah *Node*	25
Status Uji	Valid	

6.1.2 Pengujian Performa Skenario 2 Algoritme VCKNN



Gambar 6.3 Skenario 2 Algoritme VCKNN

Gambar 6.3 merupakan hasil pengujian skenario 2 pada algoritme VCKNN. Dalam algoritme VCKNN dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.623641,-7.949381]" memiliki *response time* 0.143 detik dan waktu tempuh 813 detik. Pada skenario 2 menggunakan algoritme VCKNN terdapat 25 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.2), Gambar 6.4 menjelaskan *node-node* yang dilalui pada skenario 2 menggunakan algoritme VCKNN. Tabel 6.2 menjelaskan hasil pengujian performa skenario 2 menggunakan algoritme VCKNN.



Gambar 6.4 Skenario 2 Algoritme VCKNN

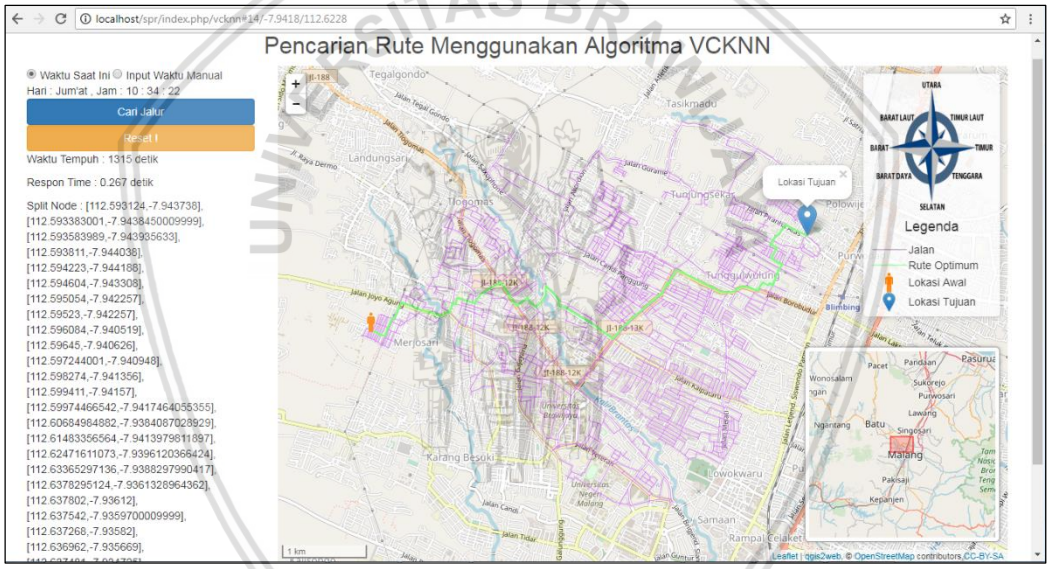
Tabel 6.2 Hasil Pengujian Performa Skenario 2 Algoritme VCKNN

Kode Pengujian	SPR-PT-02	
Nomor Skenario	Skenario 2	
Algoritme	VCKNN	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VCKNN.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.623641,-7.949381]
	Voronoi Tujuan	4
	Waktu	Hari : Jum'at , Jam : 10 : 20 : 52
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.2 Hasil Pengujian Performa Skenario 2 Algoritme VCKNN (Lanjutan)

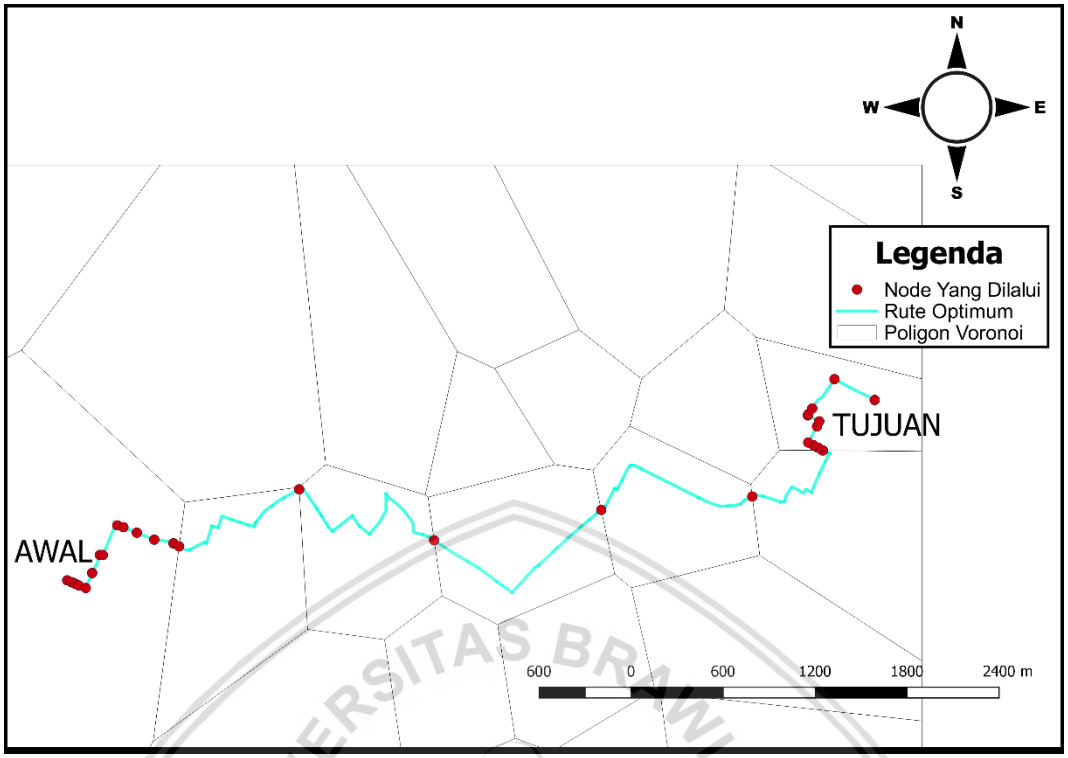
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VCKNN. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.143 detik
	Waktu Tempuh	813 detik
	Jumlah *Node*	25
Status Uji	Valid	

6.1.3 Pengujian Performa Skenario 3 Algoritme VCKNN



Gambar 6.5 Skenario 3 Algoritme VCKNN

Gambar 6.5 merupakan hasil pengujian skenario 3 pada algoritme VCKNN. Dalam algoritme VCKNN dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.640892,-7.9331800009999]" memiliki *response time* 0.267 detik dan waktu tempuh 1315 detik. Pada skenario 3 menggunakan algoritme VCKNN terdapat 30 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.3), Gambar 6.6 menjelaskan *node-node* yang dilalui pada skenario 3 menggunakan algoritme VCKNN. Tabel 6.3 menjelaskan hasil pengujian performa skenario 3 menggunakan algoritme VCKNN.



Gambar 6.6 Skenario 3 Algoritme VCKNN

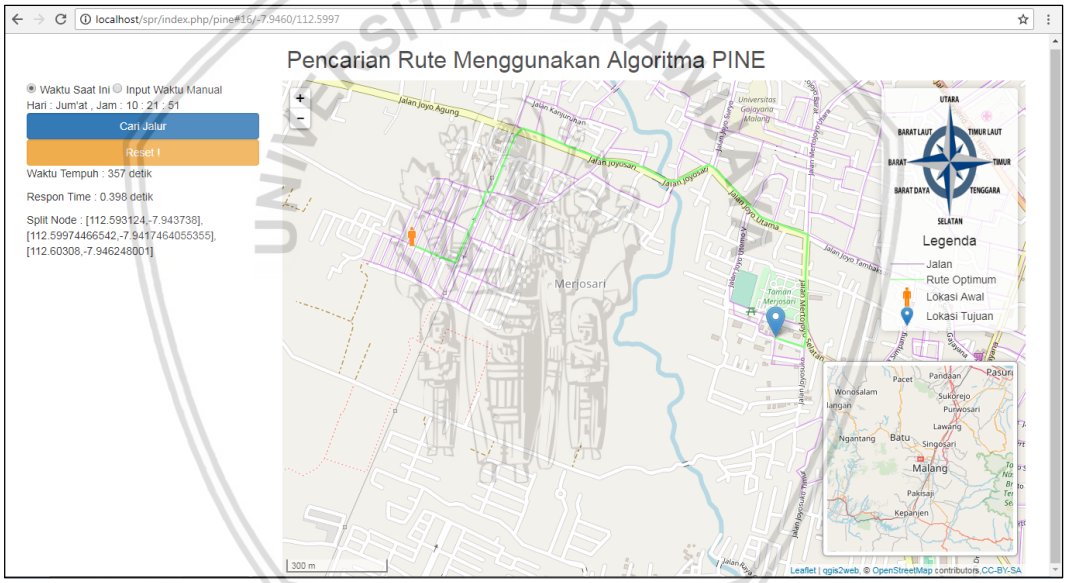
Tabel 6.3 Hasil Pengujian Performa Skenario 3 Algoritme VCKNN

Kode Pengujian	SPR-PT-03	
Nomor Skenario	Skenario 3	
Algoritme	VCKNN	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VCKNN.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.640892,-7.9331800009999]
	Voronoi Tujuan	17
	Waktu	Hari : Jum'at , Jam : 10 : 34 : 22
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.3 Hasil Pengujian Performa Skenario 3 Algoritme VCKNN (Lanjutan)

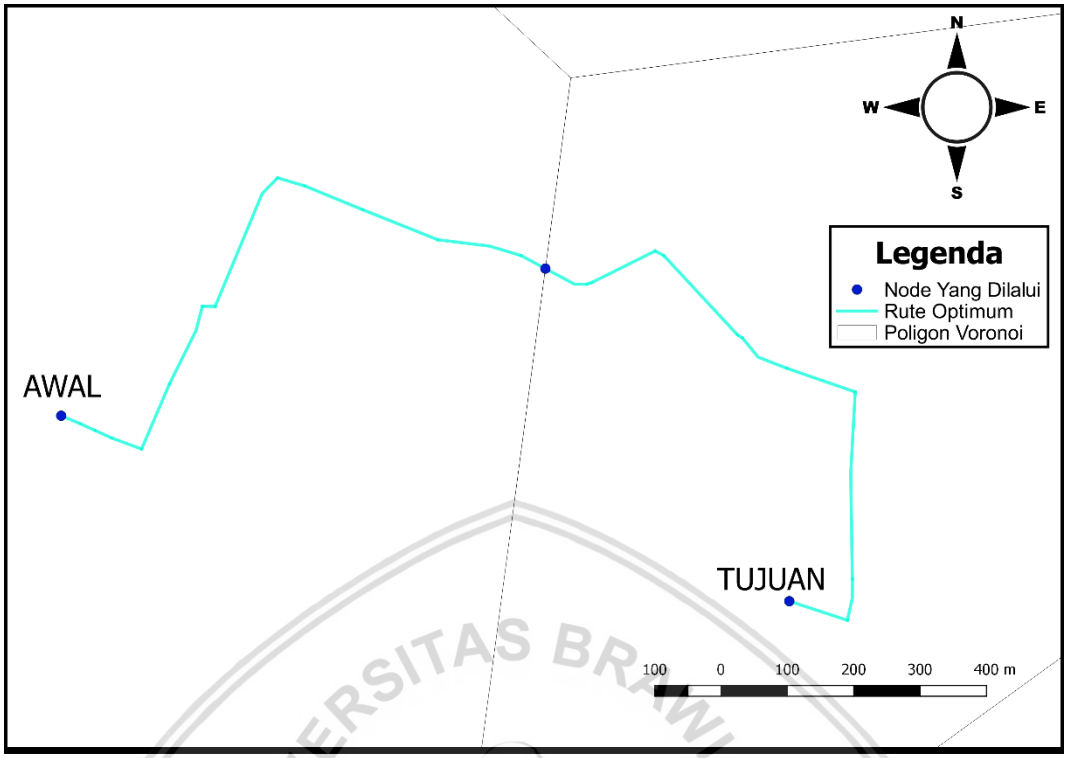
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VCKNN. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.267 detik
	Waktu Tempuh	1315 detik
	Jumlah *Node*	30
Status Uji	Valid	

6.1.4 Pengujian Performa Skenario 1 Algoritme PINE



Gambar 6.7 Skenario 1 Algoritme PINE

Gambar 6.7 merupakan hasil pengujian skenario 1 pada algoritme PINE. Dalam algoritme PINE dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.60308,-7.946248001]" memiliki *response time* 0.398 detik dan waktu tempuh 357 detik. Pada skenario 1 menggunakan algoritme PINE terdapat 3 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.4), Gambar 6.8 menjelaskan *node-node* yang dilalui pada skenario 1 menggunakan algoritme PINE. Tabel 6.4 menjelaskan hasil pengujian performa skenario 1 menggunakan algoritme PINE.



Gambar 6.8 Skenario 1 Algoritme PINE

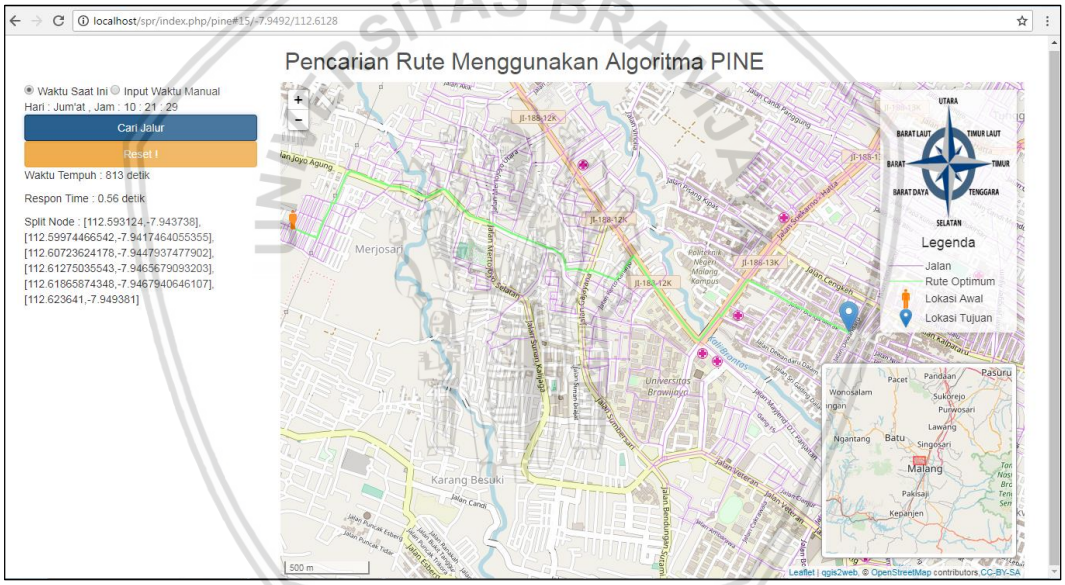
Tabel 6.4 Hasil Pengujian Performa Skenario 1 Algoritme PINE

Kode Pengujian	SPR-PT-04	
Nomor Skenario	Skenario 1	
Algoritme	PINE	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme PINE.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.60308,-7.946248001]
	Voronoi Tujuan	8
	Waktu	Hari : Jum'at , Jam : 10 : 21 : 51
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.4 Hasil Pengujian Performa Skenario 1 Algoritme PINE (Lanjutan)

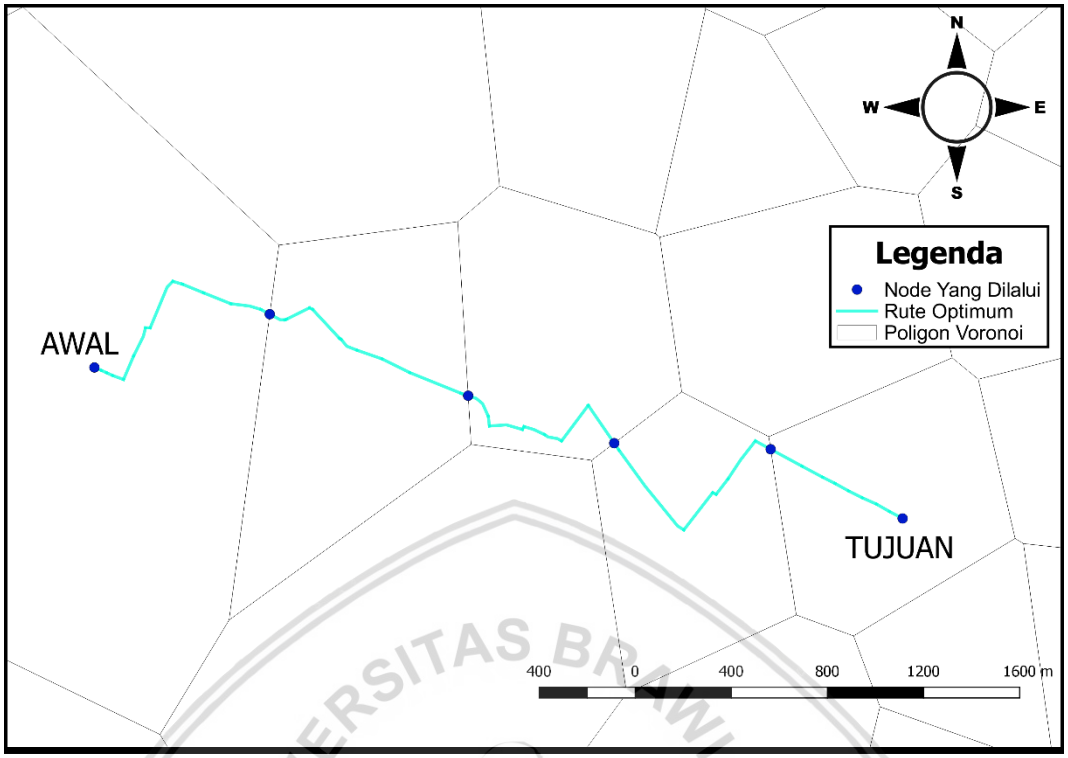
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme PINE. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.398 detik
	Waktu Tempuh	357 detik
	Jumlah *Node*	3
Status Uji	Valid	

6.1.5 Pengujian Performa Skenario 2 Algoritme PINE



Gambar 6.9 Skenario 2 Algoritme PINE

Gambar 6.9 merupakan hasil pengujian skenario 2 pada algoritme PINE. Dalam algoritme PINE dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.623641,-7.949381]" memiliki *response time* 0.56 detik dan waktu tempuh 813 detik. Pada skenario 2 menggunakan algoritme PINE terdapat 6 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.5), Gambar 6.10 menjelaskan *node-node* yang dilalui pada skenario 2 menggunakan algoritme PINE. Tabel 6.5 menjelaskan hasil pengujian performa skenario 2 menggunakan algoritme PINE.



Gambar 6.10 Skenario 2 Algoritme PINE

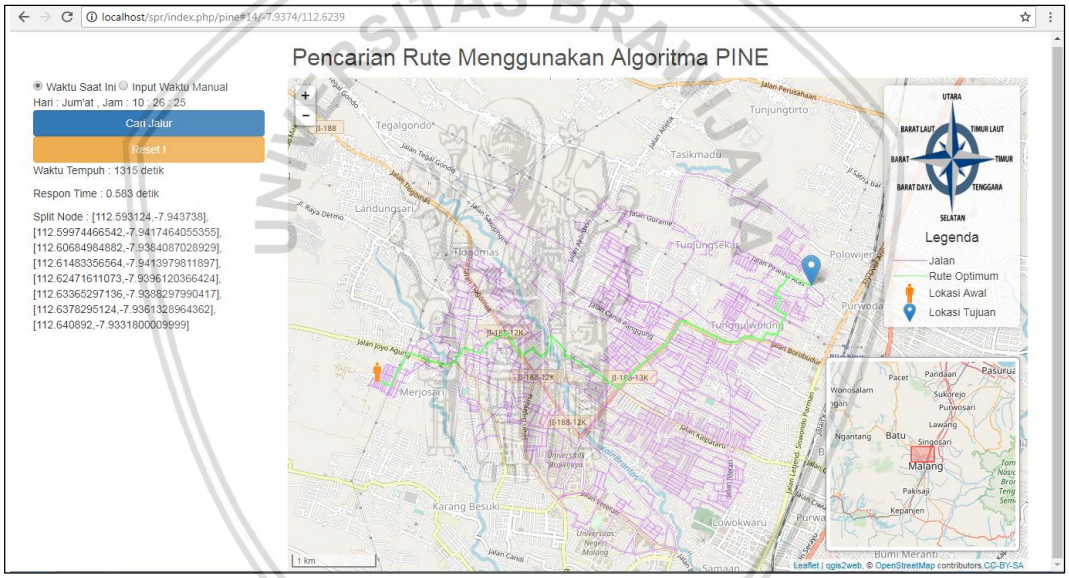
Tabel 6.5 Hasil Pengujian Performa Skenario 2 Algoritme PINE

Kode Pengujian	SPR-PT-05	
Nomor Skenario	Skenario 2	
Algoritme	PINE	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme PINE.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.623641,-7.949381]
	Voronoi Tujuan	4
	Waktu	Hari : Jum'at , Jam : 10 : 21 : 29
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.5 Hasil Pengujian Performa Skenario 2 Algoritme PINE (Lanjutan)

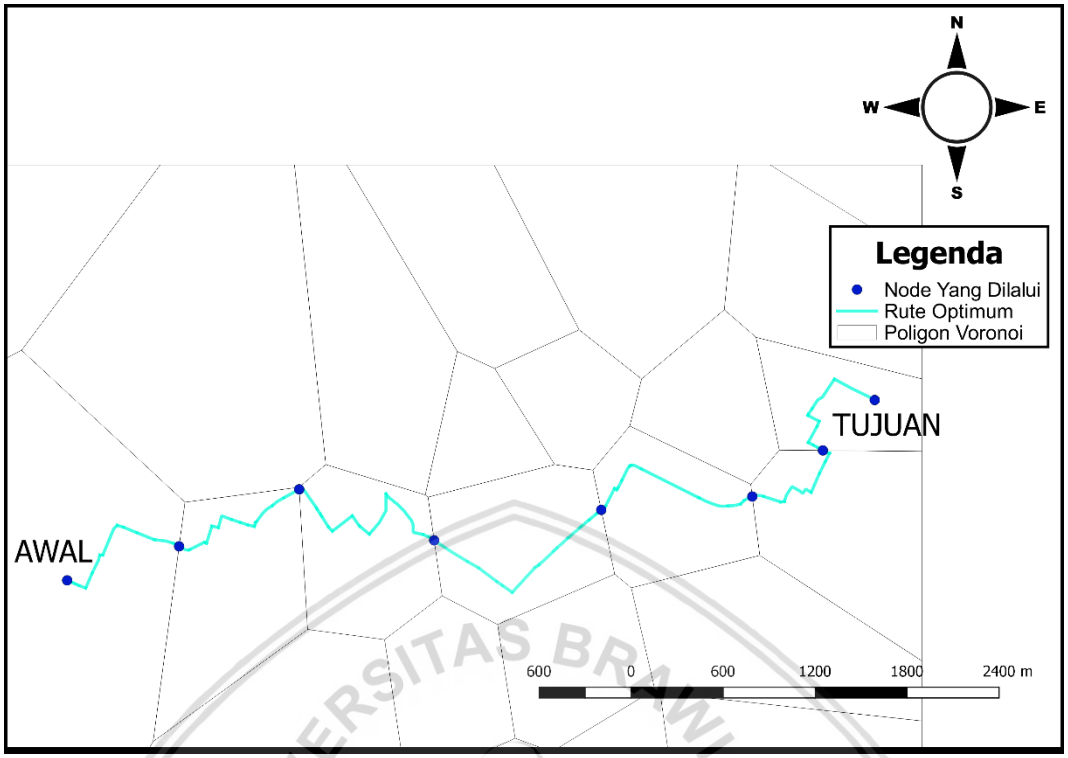
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme PINE. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.56 detik
	Waktu Tempuh	813 detik
	Jumlah *Node*	6
Status Uji	Valid	

6.1.6 Pengujian Performa Skenario 3 Algoritme PINE



Gambar 6.11 Skenario 3 Algoritme PINE

Gambar 6.11 merupakan hasil pengujian skenario 3 pada algoritme PINE. Dalam algoritme PINE dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.640892,-7.9331800009999]" memiliki *response time* 0.583 detik dan waktu tempuh 1315 detik. Pada skenario 3 menggunakan algoritme PINE terdapat 8 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.6), Gambar 6.12 menjelaskan *node-node* yang dilalui pada skenario 3 menggunakan algoritme PINE. Tabel 6.6 menjelaskan hasil pengujian performa skenario 3 menggunakan algoritme PINE.



Gambar 6.12 Skenario 3 Algoritme PINE

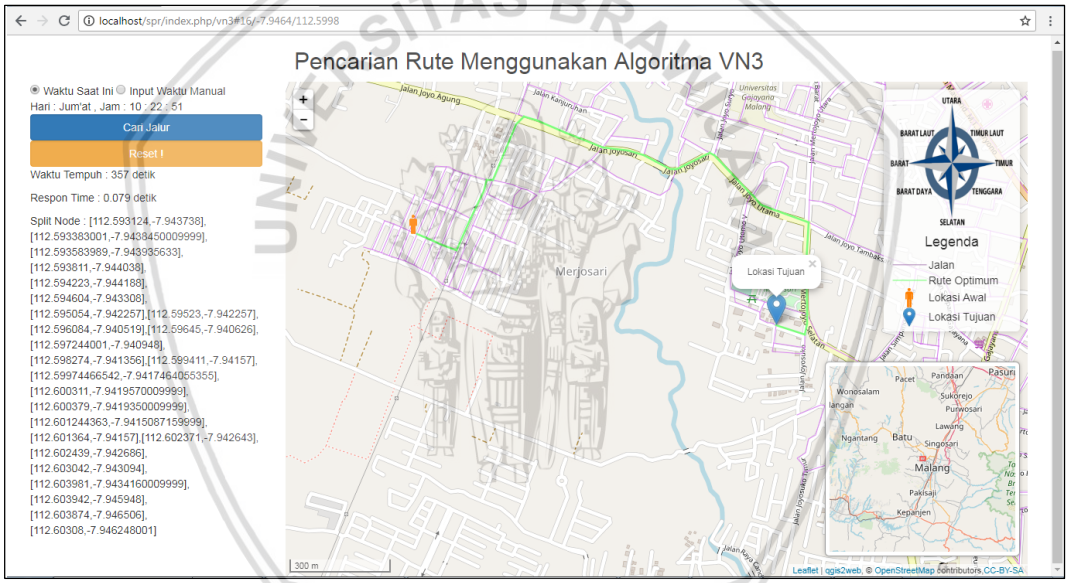
Tabel 6.6 Hasil Pengujian Performa Skenario 3 Algoritme PINE

Kode Pengujian	SPR-PT-06	
Nomor Skenario	Skenario 3	
Algoritme	PINE	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme PINE.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.640892,-7.9331800009999]
	Voronoi Tujuan	17
	Waktu	Hari : Jum'at , Jam : 10 : 26 : 25
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.6 Hasil Pengujian Performa Skenario 3 Algoritme PINE (Lanjutan)

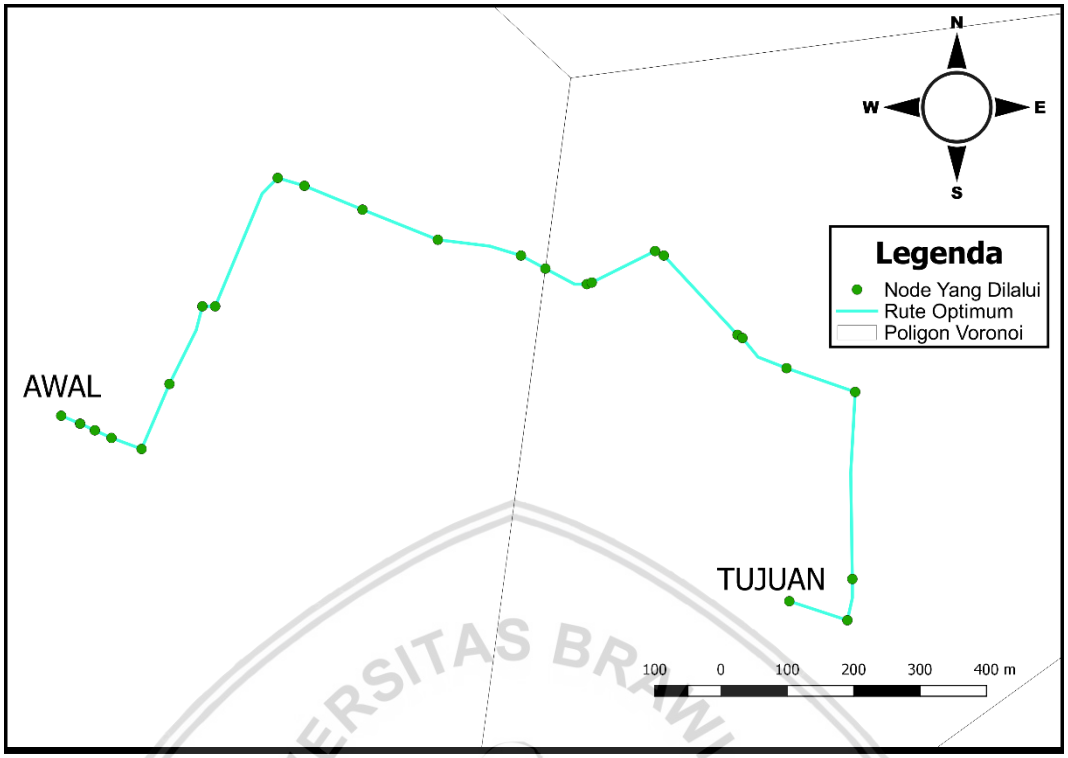
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme PINE. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.583 detik
	Waktu Tempuh	1315 detik
	Jumlah *Node*	8
Status Uji	Valid	

6.1.7 Pengujian Performa Skenario 1 Algoritme VN3



Gambar 6.13 Skenario 1 Algoritme VN3

Gambar 6.13 merupakan hasil pengujian skenario 1 pada algoritme VN3. Dalam algoritme VN3 dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.60308,-7.946248001]" memiliki *response time* 0.079 detik dan waktu tempuh 357 detik. Pada skenario 1 menggunakan algoritme VN3 terdapat 25 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.7), Gambar 6.14 menjelaskan *node-node* yang dilalui pada skenario 1 menggunakan algoritme VN3. Tabel 6.7 menjelaskan hasil pengujian performa skenario 1 menggunakan algoritme VN3.



Gambar 6.14 Skenario 1 Algoritme VN3

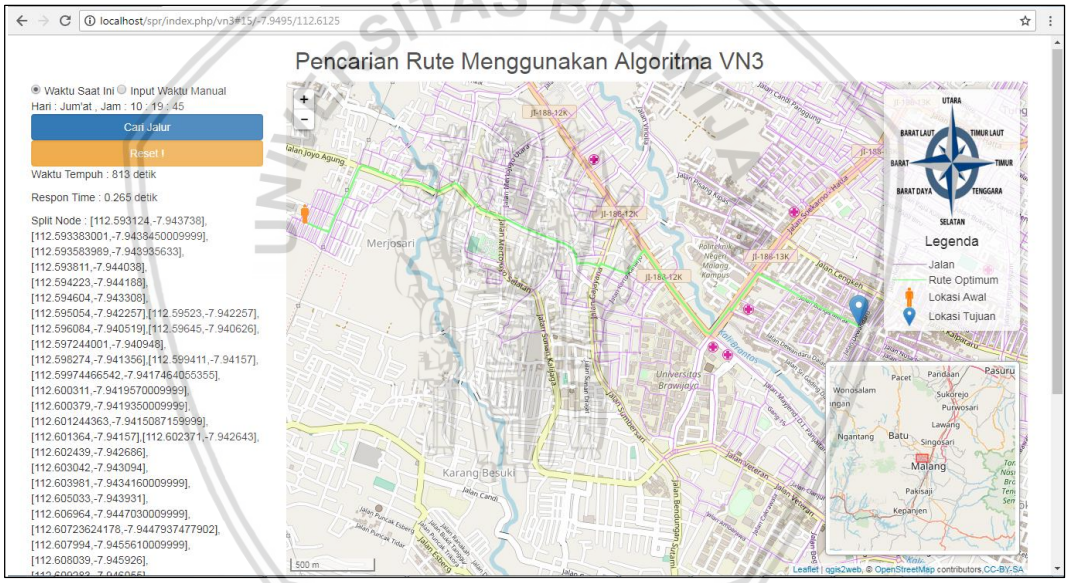
Tabel 6.7 Hasil Pengujian Performa Skenario 1 Algoritme VN3

Kode Pengujian	SPR-PT-07	
Nomor Skenario	Skenario 1	
Algoritme	VN3	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 1 menggunakan algoritme VN3.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.60308,-7.946248001]
	Voronoi Tujuan	8
	Waktu	Hari : Jum'at , Jam : 10 : 22 : 51
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.7 Hasil Pengujian Performa Skenario 1 Algoritme VN3 (Lanjutan)

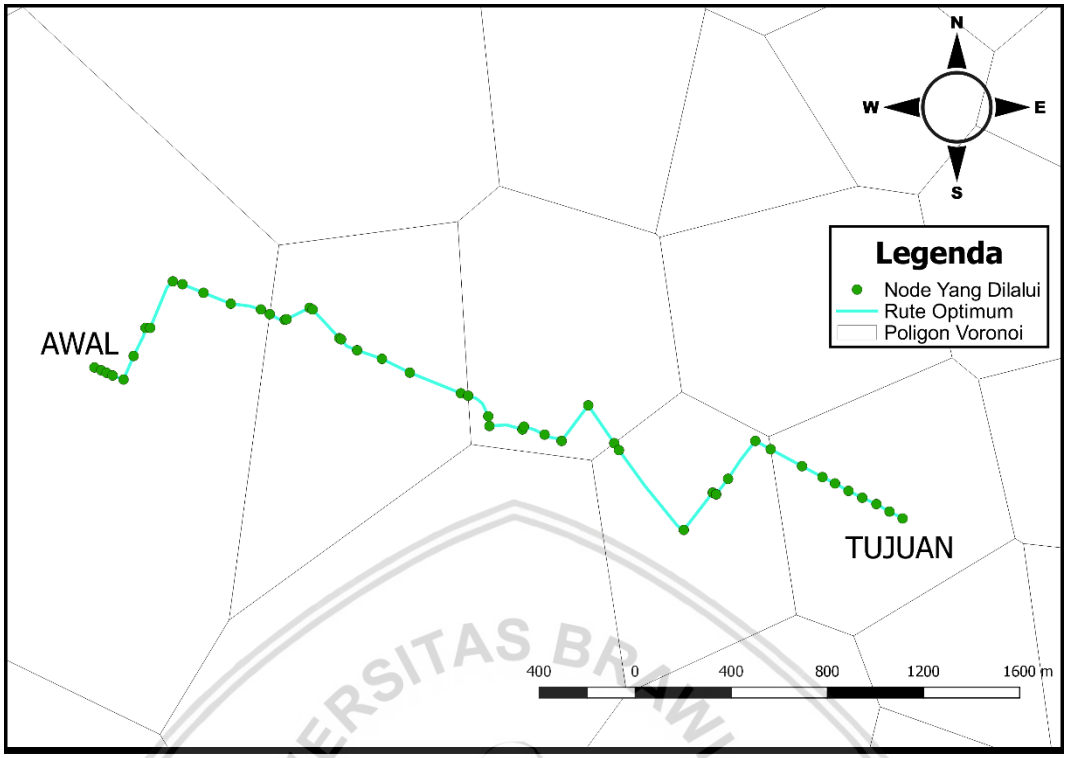
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VN3. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.079 detik
	Waktu Tempuh	357 detik
	Jumlah *Node*	25
Status Uji	Valid	

6.1.8 Pengujian Performa Skenario 2 Algoritme VN3



Gambar 6.15 Skenario 2 Algoritme VN3

Gambar 6.15 merupakan hasil pengujian skenario 2 pada algoritme VN3. Dalam algoritme VN3 dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.623641,-7.949381]" memiliki *response time* 0.265 detik dan waktu tempuh 813 detik. Pada skenario 2 menggunakan algoritme VN3 terdapat 48 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.8), Gambar 6.16 menjelaskan *node-node* yang dilalui pada skenario 2 menggunakan algoritme VN3. Tabel 6.8 menjelaskan hasil pengujian performa skenario 2 menggunakan algoritme VN3.



Gambar 6.16 Skenario 2 Algoritme VN3

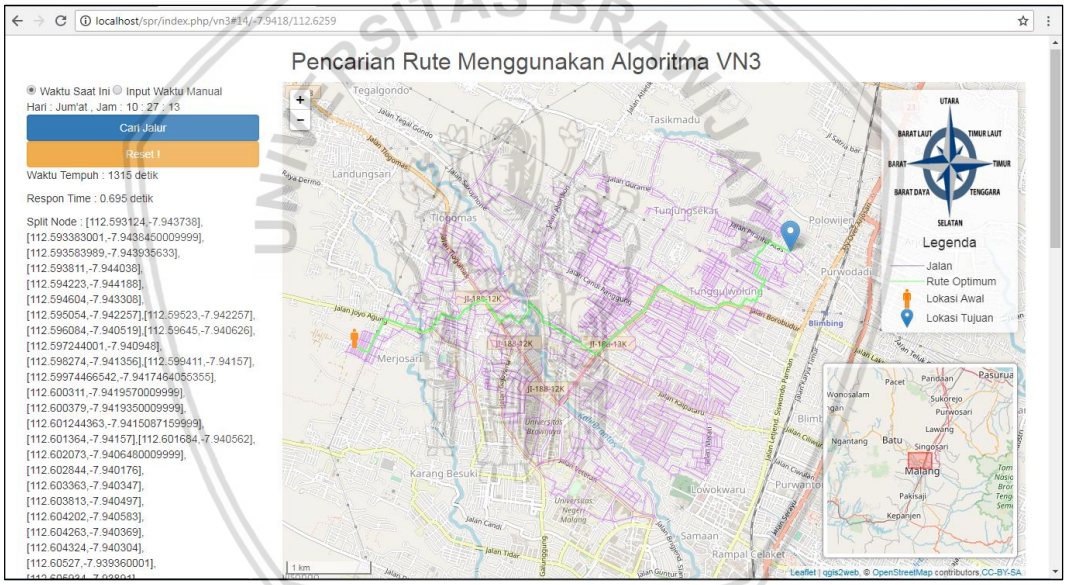
Tabel 6.8 Hasil Pengujian Performa Skenario 2 Algoritme VN3

Kode Pengujian	SPR-PT-08	
Nomor Skenario	Skenario 2	
Algoritme	VN3	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 2 menggunakan algoritme VN3.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.623641,-7.949381]
	Voronoi Tujuan	4
	Waktu	Hari : Jum'at , Jam : 10 : 19 : 45
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.8 Hasil Pengujian Performa Skenario 2 Algoritme VN3 (Lanjutan)

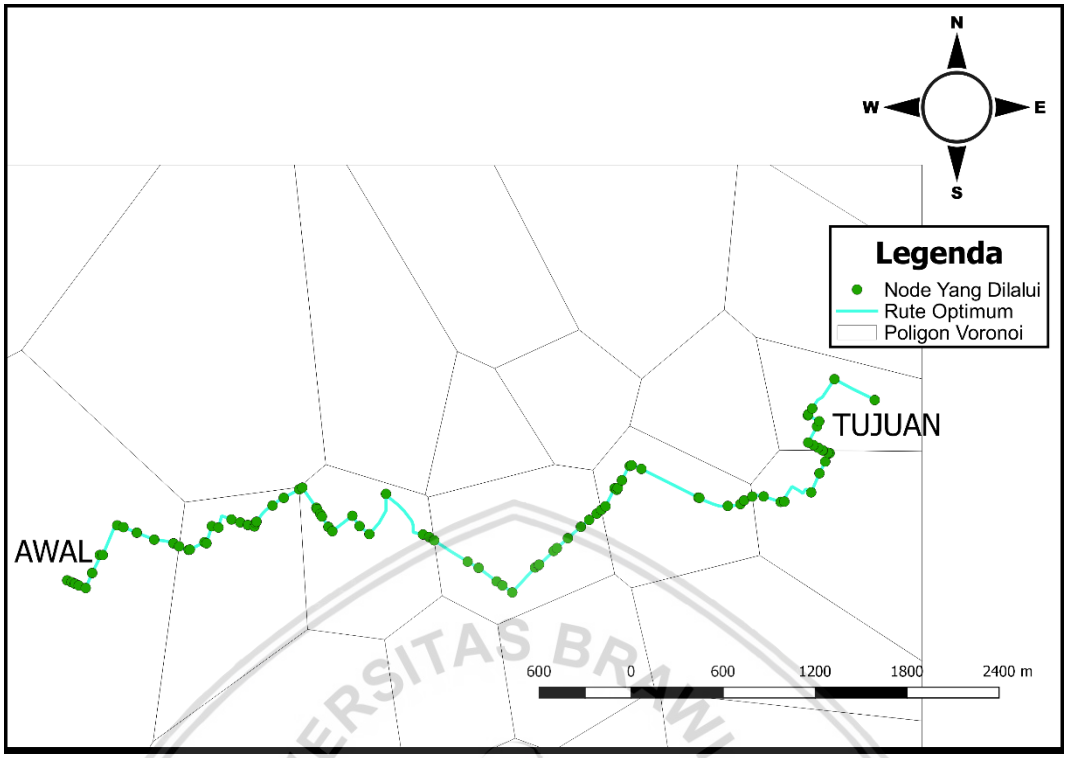
Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VN3. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.265 detik
	Waktu Tempuh	813 detik
	Jumlah *Node*	48
Status Uji	Valid	

6.1.9 Pengujian Performa Skenario 3 Algoritme VN3



Gambar 6.17 Skenario 3 Algoritme VN3

Gambar 6.17 merupakan hasil pengujian skenario 3 pada algoritme VN3. Dalam algoritme VN3 dari lokasi awal "[112.593124,-7.943738]" menuju lokasi tujuan "[112.640892,-7.9331800009999]" memiliki *response time* 0.695 detik dan waktu tempuh 1315 detik. Pada skenario 3 menggunakan algoritme VN3 terdapat 92 *node* yang dilalui dari lokasi awal ke lokasi tujuan (Lampiran A.9), Gambar 6.18 menjelaskan *node-node* yang dilalui pada skenario 3 menggunakan algoritme VN3. Tabel 6.9 menjelaskan hasil pengujian performa skenario 3 menggunakan algoritme VN3.



Gambar 6.18 Skenario 3 Algoritme VN3

Tabel 6.9 Hasil Pengujian Performa Skenario 3 Algoritme VN3

Kode Pengujian	SPR-PT-09	
Nomor Skenario	Skenario 3	
Algoritme	VN3	
Aktor	Pengguna	
Tujuan Pengujian	Pengujian untuk mendapatkan nilai *response time*, waktu tempuh dan *node* yang dilalui pada skenario 3 menggunakan algoritme VN3.	
Input Data	Lokasi Awal	[112.593124,-7.943738]
	Voronoi Awal	9
	Lokasi Tujuan	[112.640892,-7.9331800009999]
	Voronoi Tujuan	17
	Waktu	Hari : Jum'at , Jam : 10 : 27 : 13
Prosedur Uji	1. Penguji mendefinisikan waktu dengan waktu saat ini pada sistem. 2. Pengguna memasukkan lokasi awal dan lokasi tujuan. 3. Penguji menekan tombol Cari Jalur.	

Tabel 6.9 Hasil Pengujian Performa Skenario 3 Algoritme VN3 (Lanjutan)

Hasil yang Diharapkan	1. Sistem menampilkan *response time* pada algoritme VN3. 2. Sistem menampilkan waktu tempuh dari rute optimum. 3. Sistem menampilkan *node* yang dilalui pada rute optimum.	
Hasil Pengujian	*Response time*	0.695 detik
	Waktu Tempuh	1315 detik
	Jumlah *Node*	92
Status Uji	Valid	

6.2 Evaluasi

Setelah mendapat hasil dari pengujian *response time* berupa *response time* dan jumlah *node* pada masing-masing algoritme dan skenario, selanjutnya dilakukan evaluasi dengan membandingkan antara algoritme VCKNN dengan algoritme PINE, algoritme VCKNN dengan algoritme VN3 dan algoritme PINE dengan algoritme VN3 menggunakan uji *Chi-square* untuk mengetahui performa dari algoritme VCKNN, PINE dan VN3.

Tabel 6.10 merupakan hasil pengujian *response time* dari sistem pencarian rute paling optimum yang digunakan untuk uji *Chi-square*. Pada uji *Chi-square* nilai *response time*, nilai *response time* dikali 100, sehingga nilai *response time* terkecil menjadi 7,1 dan nilai terbesar menjadi 69,5, hal ini dilakukan agar frekuensi yang diamati 80% bernilai lebih dari 5 (Hidayat, 2012).

Tabel 6.10 Hasil Pengujian *Response time*

Algoritme	**Skenario**	***Response time***	***Node* yang dilalui**	**Jumlah *Node***
VCKNN	Skenario 1	0.071	25	80
	Skenario 2	0.143	25	
	Skenario 3	0.267	30	
PINE	Skenario 1	0.398	3	17
	Skenario 2	0.56	6	
	Skenario 3	0.583	8	
VN3	Skenario 1	0.079	25	165
	Skenario 2	0.265	48	
	Skenario 3	0.695	92	

6.2.1 Pengujian Statistik Algoritme VCKNN dan Algoritme PINE

Pada perbandingan algoritme VCKNN dan algoritme PINE, jumlah *node* pada algoritme PINE lebih sedikit dibandingkan dengan algoritme VCKNN, sehingga dibuat hipotesis awal dan hipotesis akhir sebagai berikut:

- H_0, algoritme PINE memiliki *response time* lebih baik dibandingkan algoritme VCKNN
- H_1, algoritme VCKNN memiliki *response time* lebih baik dibandingkan algoritme PINE

Tabel 6.11 Tabel Kontingensi *Response time* VCKNN dan PINE

Skenario	PINE	VCKNN	Total
1	39.8	7.1	46.9
2	56	14.3	70.3
3	58.3	26.7	85
Total	154.1	48.1	**202.2**

Setelah menentukan hipotesis awal dan hipotesis akhir, selanjutnya membuat tabel kontingensi dari algoritme PINE dan algoritme VCKNN (Tabel 6.11). Setelah itu mencari nilai *Chi-square* hitung dari tabel kontingensi. Tabel 6.12 adalah hasil perhitungan *Chi-square* hitung dari algoritme VCKNN dan algoritme PINE.

Tabel 6.12 Hasil *Chi-square* Hitung VCKNN dan PINE

Kolom	F0	Fh	F0-Fh	$(F0-Fh)^2$	$(F0-Fh)^2/Fh$
A	39.8	35.74327	4.056726	16.45703	0.460423014
B	7.1	11.15673	-4.05673	16.45703	1.475076643
C	56	53.57681	2.423195	5.871873	0.109597302
D	14.3	16.72319	-2.42319	5.871873	0.351121503
E	58.3	64.77992	-6.47992	41.98937	0.648185023
F	26.7	20.22008	6.479921	41.98937	2.076617713
***Chi-square* hitung**					**5.121021199**

Dari hasil uji statistik algoritme VCKNN dan algoritme PINE, didapatkan H_0 bernilai 4.605170186 dan H_1 bernilai 5.121021199, sehingga disimpulkan bahwa $H_1 > H_0$ yang berarti bahwa hipotesis awal ditolak dan hipotesis akhir diterima. Tabel 6.13 merupakan hasil dari pengujian statistik algoritme VCKNN dan algoritme PINE.

Tabel 6.13 Hasil Pengujian Statistik Algoritme VCKNN dan PINE

Kode Pengujian	SPR-ST-01	
Algoritme	VCKNN	
	PINE	
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme VCKNN dan algoritme PINE	
Prosedur Uji	1. Penguji mencari nilai *Chi-square* hitung. 2. Penguji membandingkan nilai *Chi-square* hitung dan *Chi-square* tabel.	
Jumlah *Node*	VCKNN	80
	PINE	17
Hipotesis	H_0 : Algoritme PINE memiliki *response time* lebih baik dibandingkan algoritme VCKNN	
	H_1 : Algoritme VCKNN memiliki *response time* lebih baik dibandingkan algoritme PINE	
***Response time* Algoritme VCKNN**	Skenario 1	0.071
	Skenario 2	0.143
	Skenario 3	0.267
***Response time* Algoritme PINE**	Skenario 1	0.398
	Skenario 2	0.56
	Skenario 3	0.583
***Chi-square* Tabel**	4.605170186	
***Chi-square* Hitung**	5.121021199	
Hasil	H_1 diterima dan H_0 ditolak, karena 5.121021199 > 4.605170186	
Kesimpulan	Pada perbandingan algoritme VCKNN dan algoritme PINE didapatkan hasil bahwa algoritme VCKNN memiliki performa yang lebih baik dibandingkan algoritme PINE meskipun algoritme VCKNN memiliki jumlah *node* yang lebih banyak, yang berarti pada algoritme VCKNN dan algoritme PINE jumlah *node* tidak berpengaruh dengan *response time*.	

6.2.2 Pengujian Statistik Algoritme VCKNN dan Algoritme VN3

Pada perbandingan algoritme VCKNN dan algoritme VN3, jumlah *node* pada algoritme VCKNN lebih sedikit dibandingkan dengan algoritme VN3, sehingga dibuat hipotesis awal dan hipotesis akhir sebagai berikut:

- H_0, algoritme VCKNN memiliki *response time* lebih baik dibandingkan algoritme VN3
- H_1, algoritme VN3 memiliki *response time* lebih baik dibandingkan algoritme VCKNN

Tabel 6.14 Tabel Kontingensi *Response time* VCKNN dan VN3

Skenario	VCKNN	VN3	Total
1	7.1	7.9	15
2	14.3	26.5	40.8
3	26.7	69.5	96.2
Total	48.1	103.9	**152**

Setelah menentukan hipotesis awal dan hipotesis akhir, selanjutnya membuat tabel kontingensi dari algoritme VCKNN dan algoritme VN3 (Tabel 6.14). Setelah itu mencari nilai *Chi-square* hitung dari tabel kontingensi. Tabel 6.15 adalah hasil perhitungan *Chi-square* dari algoritme VCKNN dan algoritme VN3.

Tabel 6.15 Hasil *Chi-square* Hitung VCKNN dan VN3

Kolom	F0	Fh	F0-Fh	$(F0-Fh)^2$	$(F0-Fh)^2/Fh$
A	7.1	4.746711	2.353289	5.537971	1.166696666
B	7.9	10.25329	-2.35329	5.537971	0.540116551
C	14.3	12.91105	1.388947	1.929175	0.149420411
D	26.5	27.88895	-1.38895	1.929175	0.069173453
E	26.7	30.44224	-3.74224	14.00434	0.460029815
F	69.5	65.75776	3.742237	14.00434	0.212968567
***Chi-square* hitung**					**2.598405463**

Dari hasil uji statistik algoritme VCKNN dan algoritme VN3, didapatkan H_0 bernilai 4.605170186 dan H_1 bernilai 2.598405463, sehingga disimpulkan bahwa $H_1<H_0$ yang berarti bahwa hipotesis awal diterima dan hipotesis akhir ditolak. Tabel 6.16 merupakan hasil dari pengujian statistik algoritme VCKNN dan algoritme VN3.

Tabel 6.16 Hasil Pengujian Statistik Algoritme VCKNN dan VN3

Kode Pengujian	SPR-ST-02	
Algoritme	VCKNN	
	VN3	
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme VCKNN dan algoritme VN3	
Prosedur Uji	1. Penguji mencari nilai *Chi-square* hitung. 2. Penguji membandingkan nilai *Chi-square* hitung dan *Chi-square* tabel.	
Jumlah *Node*	VCKNN	80
	VN3	165
Hipotesis	H_0 : Algoritme VCKNN memiliki *response time* lebih baik dibandingkan algoritme VN3	
	H_1 : Algoritme VN3 memiliki *response time* lebih baik dibandingkan algoritme VCKNN	
***Response time* Algoritme VCKNN**	Skenario 1	0.071
	Skenario 2	0.143
	Skenario 3	0.267
***Response time* Algoritme VN3**	Skenario 1	0.079
	Skenario 2	0.265
	Skenario 3	0.695
***Chi-square* Tabel**	4.605170186	
***Chi-square* Hitung**	2.598405463	
Hasil	H_1 ditolak dan H_0 diterima, karena 2.598405463 < 4.605170186	
Kesimpulan	Pada perbandingan algoritme VCKNN dan algoritme VN3 didapatkan hasil bahwa algoritme VCKNN memiliki performa yang lebih baik dibandingkan algoritme VN3 yang berarti pada algoritme VCKNN dan algoritme VN3 jumlah *node* berpengaruh dengan *response time*.	

6.2.3 Pengujian Statistik Algoritme PINE dan Algoritme VN3

Pada perbandingan algoritme PINE dan algoritme VN3, jumlah *node* pada algoritme PINE lebih sedikit dibandingkan dengan algoritme VN3, sehingga dibuat hipotesis awal dan hipotesis akhir sebagai berikut:

- H_0, algoritme PINE memiliki *response time* lebih baik dibandingkan algoritme VN3
- H_1, algoritme VN3 memiliki *response time* lebih baik dibandingkan algoritme PINE

Tabel 6.17 Tabel Kontingensi *Response time* PINE dan VN3

Skenario	PINE	VN3	Total
1	39.8	7.9	47.7
2	56	26.5	82.5
3	58.3	69.5	127.8
Total	154.1	103.9	**258**

Setelah menentukan hipotesis awal dan hipotesis akhir, selanjutnya membuat tabel kontingensi dari algoritme PINE dan algoritme VN3 (Tabel 6.17). Setelah itu mencari nilai *Chi-square* hitung dari tabel kontingensi. Tabel 6.18 adalah hasil perhitungan *Chi-square* dari algoritme PINE dan algoritme VN3.

Tabel 6.18 Hasil *Chi-square* Hitung PINE dan VN3

Kolom	F0	Fh	F0-Fh	$(F0-Fh)^2$	$(F0-Fh)^2/Fh$
A	39.8	28.49058	11.30942	127.9029	4.489306392
B	7.9	19.20942	-11.3094	127.9029	6.658345669
C	56	49.27616	6.723837	45.20999	0.917481887
D	26.5	33.22384	-6.72384	45.20999	1.360769574
E	58.3	76.33326	-18.0333	325.1983	4.260244264
F	69.5	51.46674	18.03326	325.1983	6.318610598
***Chi-square* hitung**					**24.00475838**

Dari hasil uji statistik algoritme PINE dan algoritme VN3, didapatkan H_0 bernilai 4.605170186 dan H_1 bernilai 24.00475838, sehingga disimpulkan bahwa $H_1 > H_0$ yang berarti bahwa hipotesis awal ditolak dan hipotesis akhir diterima. Tabel 6.19 merupakan hasil dari pengujian statistik algoritme PINE dan algoritme VN3.

Tabel 6.19 Hasil Pengujian Statistik Algoritme PINE dan VN3

Kode Pengujian	SPR-ST-03
Algoritme	PINE
	VN3
Tujuan Pengujian	Pengujian dilakukan untuk mengetahui perbandingan performa antara algoritme PINE dan algoritme VN3

Tabel 6.19 Hasil Pengujian Statistik Algoritme PINE dan VN3 (Lanjutan)

Prosedur Uji	1. Penguji mencari nilai *Chi-square* hitung. 2. Penguji membandingkan nilai *Chi-square* hitung dan *Chi-square* tabel.	
Jumlah *Node*	PINE	17
	VN3	165
Hipotesis	H_0 : Algoritme PINE memiliki *response time* lebih baik dibandingkan algoritme VN3	
	H_1 : Algoritme VN3 memiliki *response time* lebih baik dibandingkan algoritme PINE	
***Response time* Algoritme PINE**	Skenario 1	0.398
	Skenario 2	0.56
	Skenario 3	0.583
***Response time* Algoritme VN3**	Skenario 1	0.079
	Skenario 2	0.265
	Skenario 3	0.695
***Chi-square* Tabel**	4.605170186	
***Chi-square* Hitung**	24.00475838	
Hasil	H_1 diterima dan H_0 ditolak, karena 24.00475838 > 4.605170186	
Kesimpulan	Pada perbandingan algoritme PINE dan algoritme VN3 didapatkan hasil bahwa algoritme VN3 memiliki performa yang lebih baik dibandingkan algoritme PINE meskipun algoritme VN3 memiliki jumlah *node* yang lebih banyak, yang berarti pada algoritme VN3 dan algoritme PINE jumlah *node* tidak berpengaruh dengan *response time*.	

BAB 7 PENUTUP

7.1 Kesimpulan

Dari penelitian yang dilakukan, pengembangan sistem pencarian rute paling optimum menggunakan algoritme VCKNN, PINE dan VN3 didapatkan kesimpulan sebagai berikut:

1. Dengan sistem pencarian rute paling optimum, dapat diketahui *response time* pada algoritme VCKNN, PINE dan VN3 dalam 3 skenario. Pada algoritme VCKNN memiliki *response time* 0.071 detik pada skenario 1, 0.143 detik pada skenario 2 dan 0.267 detik pada skenario 3. Pada algoritme PINE memiiki *response time* 0.398 detik pada skenario 1, 0.56 detik pada skenario 2 dan 0.583 detik pada skenario 3. Pada algoritme VN3 memiliki *response time* 0.079 pada skenario 1, 0.265 pada skenario 2 dan 0.695 pada skenario 3. Sehingga dapat disimpulkan algoritme dengan *response time* tercepat dalam pencarian rute pada skenario 1 adalah VCKNN setelah itu VN3 dan terakhir PINE, pada skenario 2 adalah VCKNN setelah itu VN3 dan terakhir PINE, pada skenario 3 adalah VCKNN setelah itu PINE dan terakhir VN3. Dari 3 skenario algoritme dengan *response time* tercepat dalam pencarian rute paling optimum adalah algoritme VCKNN.
2. Pada algoritme VCKNN terdapat 25 *node* pada skenario 1, 25 *node* pada skenario 2, 30 *node* pada skenario 3 dengan jumlah 80 *node*. Pada algoritme PINE terdapat 3 *node* pada skenario 1, 6 *node* pada skenario 2, 8 *node* pada skenario 3 dengan jumlah 17 *node*. Pada algoritme VN3 terdapat 25 *node* pada skenario 1, 48 *node* pada skenario 2, 92 *node* pada skenario 3 dengan jumlah *node* 165 *node*. Sehingga dapat disimpulkan algoritme yang paling sedikit memiliki jumlah *node* yang dilalui pada skenario 1 adalah PINE setelah itu VCKNN dan VN3 dengan jumlah *node* yang sama, pada skenario 2 adalah PINE setelah itu VCKNN dan terakhir VN3, pada skenario 3 adalah PINE setelah itu VCKNN dan terakhir VN3. Dari 3 skenario algoritme dengan jumlah *node* yang dilalui paling sedikit adalah algoritme PINE.
3. Berdasarkan hasil pengujian statistik *Chi-square*, setiap perbandingan algoritme dibagi menjadi 3, yaitu algoritme VCKNN dengan algoritme PINE, algoritme VCKNN dengan algoritme VN3 dan algoritme PINE dengan VN3. Pada algoritme VCKNN dan algoritme PINE didapatkan hasil bahwa jumlah *node* yang dilalui tidak mempengaruhi *response time*. Pada algoritme VCKNN dan VN3 didapatkan hasil bahwa jumlah *node* yang dilalui mempengaruhi *response time*. Pada algoritme PINE dan VN3 didapatkan hasil bahwa jumlah *node* yang dilalui tidak mempengaruhi *response time*.
4. Berdasarkan hasil pengujian statistik *Chi-square*, didapatkan hasil yang berbeda. Pada perbandingan algoritme VCKNN dan algoritme VN3 jumlah *node* mempengaruhi *response time* karena pada algoritme VCKNN menggunakan data *move interval* pada *voronoi* tetangga serta jaringan jalan *voronoi* pada *voronoi* awal dan *voronoi* tujuan, sedangkan algoritme VN3

menggunakan data jaringan jalan *voronoi* di semua *voronoi* sehingga jumlah *node* yang dilalui lebih banyak dari algoritme VCKNN, hal tersebut mempengaruhi *response time*. Sedangkan pada perbandingan algoritme VCKNN dengan algoritme PINE dan algoritme PINE dengan algoritme VN3, jumlah *node* yang dilalui tidak mempengaruhi *response time*, hal ini dikarenakan pada algoritme PINE harus mencari *move interval* terlebih dahulu pada *voronoi* awal dan *voronoi* tujuan, sehingga menambah proses dalam algoritme PINE yang berdampak pada *response time*. Sehingga pada algoritme VCKNN dan PINE jumlah *node* lebih sedikit dibandingkan VN3 karena menggunakan data *move interval* dan algoritme VCKNN memiliki *response time* lebih cepat dari PINE karena tidak ada proses mencari *move interval* terlebih dahulu.

7.2 Saran

Pada sub bab saran menjelaskan saran yang diberikan untuk pengembangan lanjut sistem pencarian rute paling optimum diantaranya adalah :

1. Diberikan fitur untuk mengubah data jaringan jalan pada aplikasi.
2. Data waktu tempuh setiap segmen didapatkan secara langsung dari pengguna pada waktu sebenarnya menggunakan teknologi *GPS* pada *smartphone*.
3. Dilakukan penelitian yang lebih mendalam tentang penentuan *interest point*, seperti membandingkan penggunaan rumah sakit dan lampu merah sebagai penenentuan *interest point* sehingga algoritme yang digunakan menjadi lebih baik dalam pembuatan *move iterval*.
4. Dilakukan analisis terhadap penyimpanan data, karena *move interval* disimpan di *database*, sehingga algoritme VCKNN dan algoritme PINE membutuhkan penyimpanan yang lebih besar dibandingkan dengan algoritme VN3.

DAFTAR PUSTAKA

Alhadar, A., 2011.'Analisis Kinerja Jalan Dalam Upaya Mengatasi Kemacetan Lalu Lintas Pada Ruas Simpang Bersinyal Di Kota Palu', *Jurnal Sipil Mesin Arsitektur Elektro*, vol. 9, no. 4,pp. 327 - 336

Aris, A., Ashar , K., 2013.'Analisis Dampak Sosial Ekonomi Pengguna Jalan Akibat Kemacetan Lalulintas (Studi Kasus Area Sekitar Universitas Brawijaya Malang)', *Jurnal Ilmiah Mahasiswa FEB*, vol. 1, no. 2

Aronoff, S., 1989. *Geographic Information Systems : A Management Perspective*, WDL Publication, Otawa, Canada.

Arum, D., A., P., 2014.'Optimalisasi Penyidik Unit Reserse Dalam Menangani Pencurian Dengan Kekerasan', *Jurnal Mahasiswa Fakultas Hukum*

Badan Pusat Statistik, 2010 , Jumlah dan Distribusi Penduduk Tersedia di <http://sp2010.bps.go.id/index.php/site/index> [Diakses pada 12 April 2017]

Badan Pusat Statistik, 2014 , Jumlah Kendaraan Bermotor (Unit) Tersedia di <https://data.go.id/dataset/jumlah-kendaraan-bermotor-unit> [Diakses pada 21 Februari 2017]

Budihartono, E., 2016.'Penerapan Algoritma Dijkstra Untuk Sistem Pendukung Keputusan Bagi Penentuan Jalur Terpendek Pengiriman Paket Barang Pada Traver', *Prosiding Seminar Nasional IPTEK Terapan*, vol. 1, no. 1, pp. 69-78

Chang, K., T., 2002. *Intoduction to Geographic Information System* . Mc Graw. Hill.

Ekawati, N., N., Soeaidy, S., M., Ribawanto, H., 2014.'Kajian Dampak Pengembangan Pembangunan Kota Malang Terhadap Kemacetan Lalu Lintas (Studi pada Dinas Perhubungan Kota Malang)', *Jurnal Administrasi Publik*, vol. 2, no. 1, pp. 129-133

Esri. 2002. *ArcGIS Brings Topology to the Geodatabase.* Tersedia di <http://www.esri.com/news/arcnews/summer02articles/arcgis-brings-topology.html> [Diakses pada 21 Februari 2017]

Harahap, M., K., Khairina, N., 2017.'Pencarian Jalur Terpendek dengan Algoritma Dijkstra', *Jurnal & Penelitian Teknik Informatika,* vol. 2, no. 2, pp. 18-23

Hidayat, A., 2012, Tutorial Rumus Chi Square Dan Metode Hitung. Tersedia di <https://www.statistikian.com/2012/11/rumus-chi-square.html> [Diakses pada 6 Juli 2018]

Khasanah, A., K., 2015. *Pengembangan dan Analisis Kualitas Berdasarkan ISO 9126 Aplikasi Pendeteksi Gaya Belajar Model VAK (Visual, Auditorial, Kinestetik) Berbasis Web*. S1. Universitas Negeri Yogyakarta.

Longley, P.A et. al. 2001. *Geographic Information System and Science*. John Wiley & Sons. New York.

Molyneaux, I., 2015. *The Art of Application Performance Testing(SECOND EDITION)*. Sebastopol: O'Reilly.

Patriandini, A., Suharyadi, R., Kadyarsi, I., 2013.'Kajian Tingkat Kemacetan Lalu-Lintas Dengan Memanfaatkan Citra Quickbird Dan Sistem Informasi Geografis Di Sebagian Ruas Jalan Kota Tegal', *Jurnal Bumi Indonesia*, vol. 2, no. 1, pp. 153-163

Peuquet, D., J., 2017. *Time in GIS and geographical databases* ,pp : 91-103

Presman, R., S., 2010. *Software Engineering A Practitioner's Approach* . New York: McGraw-Hill.

Rachmawati, A., 2010.'Aplikasi SIG (Sistem Informasi Geografis) Untuk Evaluasi Sistem Jaringan Drainase Di Sub DAS Lowokwaru Kota Malang', *Jurnal Rekayasa Sipil*, vol. 4, no.2, pp. 111-123

Raharja, H.,S., 2017, Panduan Uji Chi Square Untuk Kasus Satu Sampel. Tersedia di <https://statmat.id/panduan-uji-chi-square-kasus-satu-sampel> [Diakses pada 6 Juli 2018]

Safar, M., 2005.*'K nearest neighbor search in navigation system', Mobile Information System*', vol .1, pp 207-224

Songnian, L., S., D., B., V., 2011. *Advances in Web-based GIS, Mapping Services and Applications*. London: CRC Press/Balkema.

Sukamto, R., A., Shalahuddin, M., 2013. *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Object*. Bandung : INFORMATIKA

Wiyono, S., 2012.'Penggunakan Sistem Dinamik Dalam Manajemen Transportasi Untuk Mengatasi Kemacetan Di Daerah Perkotaan', *Jurnal Transportasi*, vol. 12, no. 1, pp. 1-10

Zhao, G., Xuan, K., Rahayu, W., Taniar, D., Safar, M., Gavrilova, M., L., Srinivasan, B., 2011. '*Voronoi-Based Continuous k Nearest Neighbor Search in Mobile Navigation*', *IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS* S, vol. 58, no. 6, hh. 2247-2257

LAMPIRAN A *NODE* YANG DILALUI

A.1 Tabel *Node* Yang Dilalui Skenario 1 Algoritme VCKNN

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.593383001,-7.9438450009999]
3	[112.593583989,-7.943935633]
4	[112.593811,-7.944038]
5	[112.594223,-7.944188]
6	[112.594604,-7.943308]
7	[112.595054,-7.942257]
8	[112.59523,-7.942257]
9	[112.596084,-7.940519]
10	[112.59645,-7.940626]
11	[112.597244001,-7.940948]
12	[112.598274,-7.941356]
13	[112.599411,-7.94157]
14	[112.59974466542,-7.9417464055355]
15	[112.600311,-7.9419570009999]
16	[112.600379,-7.9419350009999]
17	[112.601244363,-7.9415087159999]
18	[112.601364,-7.94157]
19	[112.602371,-7.942643]
20	[112.602439,-7.942686]
21	[112.603042,-7.943094]
22	[112.603981,-7.9434160009999]
23	[112.603942,-7.945948]
24	[112.603874,-7.946506]
25	[112.60308,-7.946248001]

A.2 Tabel *Node* Yang Dilalui Skenario 2 Algoritme VCKNN

Nomor	*Node*
1	[112.593124,-7.943738]

2	[112.593383001,-7.9438450009999]
3	[112.593583989,-7.943935633]
4	[112.593811,-7.944038]
5	[112.594223,-7.944188]
6	[112.594604,-7.943308]
7	[112.595054,-7.942257]
8	[112.59523,-7.942257]
9	[112.596084,-7.940519]
10	[112.59645,-7.940626]
11	[112.597244001,-7.940948]
12	[112.598274,-7.941356]
13	[112.599411,-7.94157]
14	[112.59974466542,-7.9417464055355]
15	[112.60723624178,-7.9447937477902]
16	[112.61275035543,-7.9465679093203]
17	[112.61865874348,-7.9467940646107]
18	[112.619842,-7.947428]
19	[112.620613,-7.947836]
20	[112.621086,-7.948072]
21	[112.621597,-7.9483510009999]
22	[112.622116,-7.948608001]
23	[112.62265,-7.948844]
24	[112.623146,-7.949123]
25	[112.623641,-7.949381]

A.3 Tabel *Node* Yang Dilalui Skenario 3 Algoritme VCKNN

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.593383001,-7.9438450009999]
3	[112.593583989,-7.943935633]
4	[112.593811,-7.944038]
5	[112.594223,-7.944188]
6	[112.594604,-7.943308]

7	[112.595054,-7.942257]
8	[112.59523,-7.942257]
9	[112.596084,-7.940519]
10	[112.59645,-7.940626]
11	[112.597244001,-7.940948]
12	[112.598274,-7.941356]
13	[112.599411,-7.94157]
14	[112.59974466542,-7.9417464055355]
15	[112.60684984882,-7.9384087028929]
16	[112.61483356564,-7.9413979811897]
17	[112.62471611073,-7.9396120366424]
18	[112.63365297136,-7.9388297990417]
19	[112.6378295124,-7.9361328964362]
20	[112.637802,-7.93612]
21	[112.637542,-7.9359700009999]
22	[112.637268,-7.93582]
23	[112.636962,-7.935669]
24	[112.637481,-7.934725]
25	[112.637611,-7.934425001]
26	[112.63694,-7.934082]
27	[112.636962,-7.934017001]
28	[112.637199,-7.933674]
29	[112.638511,-7.931957]
30	[112.640892,-7.9331800009999]

A.4 Tabel *Node* Yang Dilalui Skenario 1 Algoritme PINE

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.59974466542,-7.9417464055355]
3	[112.60308,-7.946248001]

A.5 Tabel *Node* Yang Dilalui Skenario 2 Algoritme PINE

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.59974466542,-7.9417464055355]
3	[112.60723624178,-7.9447937477902]
4	[112.61275035543,-7.9465679093203]
5	[112.61865874348,-7.9467940646107]
6	[112.623641,-7.949381]

A.6 Tabel *Node* Yang Dilalui Skenario 3 Algoritme PINE

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.59974466542,-7.9417464055355]
3	[112.60684984882,-7.9384087028929]
4	[112.61483356564,-7.9413979811897]
5	[112.62471611073,-7.9396120366424]
6	[112.63365297136,-7.9388297990417]
7	[112.6378295124,-7.9361328964362]
8	[112.640892,-7.9331800009999]

A.7 Tabel *Node* Yang Dilalui Skenario 1 Algoritme VN3

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.593383001,-7.9438450009999]
3	[112.593583989,-7.943935633]
4	[112.593811,-7.944038]
5	[112.594223,-7.944188]
6	[112.594604,-7.943308]
7	[112.595054,-7.942257]
8	[112.59523,-7.942257]
9	[112.596084,-7.940519]
10	[112.59645,-7.940626]
11	[112.597244001,-7.940948]

12	[112.598274,-7.941356]
13	[112.599411,-7.94157]
14	[112.59974466542,-7.9417464055355]
15	[112.600311,-7.9419570009999]
16	[112.600379,-7.9419350009999]
17	[112.601244363,-7.9415087159999]
18	[112.601364,-7.94157]
19	[112.602371,-7.942643]
20	[112.602439,-7.942686]
21	[112.603042,-7.943094]
22	[112.603981,-7.9434160009999]
23	[112.603942,-7.945948]
24	[112.603874,-7.946506]
25	[112.60308,-7.946248001]

A.8 Tabel *Node* Yang Dilalui Skenario 2 Algoritme VN3

Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.593383001,-7.9438450009999]
3	[112.593583989,-7.943935633]
4	[112.593811,-7.944038]
5	[112.594223,-7.944188]
6	[112.594604,-7.943308]
7	[112.595054,-7.942257]
8	[112.59523,-7.942257]
9	[112.596084,-7.940519]
10	[112.59645,-7.940626]
11	[112.597244001,-7.940948]
12	[112.598274,-7.941356]
13	[112.599411,-7.94157]
14	[112.59974466542,-7.9417464055355]
15	[112.600311,-7.9419570009999]
16	[112.600379,-7.9419350009999]

17	[112.601244363,-7.9415087159999]
18	[112.601364,-7.94157]
19	[112.602371,-7.942643]
20	[112.602439,-7.942686]
21	[112.603042,-7.943094]
22	[112.603981,-7.9434160009999]
23	[112.605033,-7.943931]
24	[112.606964,-7.9447030009999]
25	[112.60723624178,-7.9447937477902]
26	[112.607994,-7.9455610009999]
27	[112.608039,-7.945926]
28	[112.609283,-7.946055]
29	[112.609344,-7.945948]
30	[112.610122,-7.946248001]
31	[112.610763,-7.946484]
32	[112.61177,-7.9451540009999]
33	[112.61275035543,-7.9465679093203]
34	[112.61293,-7.946827]
35	[112.615379,-7.94981]
36	[112.61647,-7.948415]
37	[112.6166,-7.94848]
38	[112.61705,-7.9479]
39	[112.61808,-7.946484]
40	[112.61865874348,-7.9467940646107]
41	[112.619842,-7.947428]
42	[112.620613,-7.947836]
43	[112.621086,-7.948072]
44	[112.621597,-7.9483510009999]
45	[112.622116,-7.948608001]
46	[112.62265,-7.948844]
47	[112.623146,-7.949123]
48	[112.623641,-7.949381]

A.9 Tabel *Node* Yang Dilalui Skenario 3 Algoritme VN3

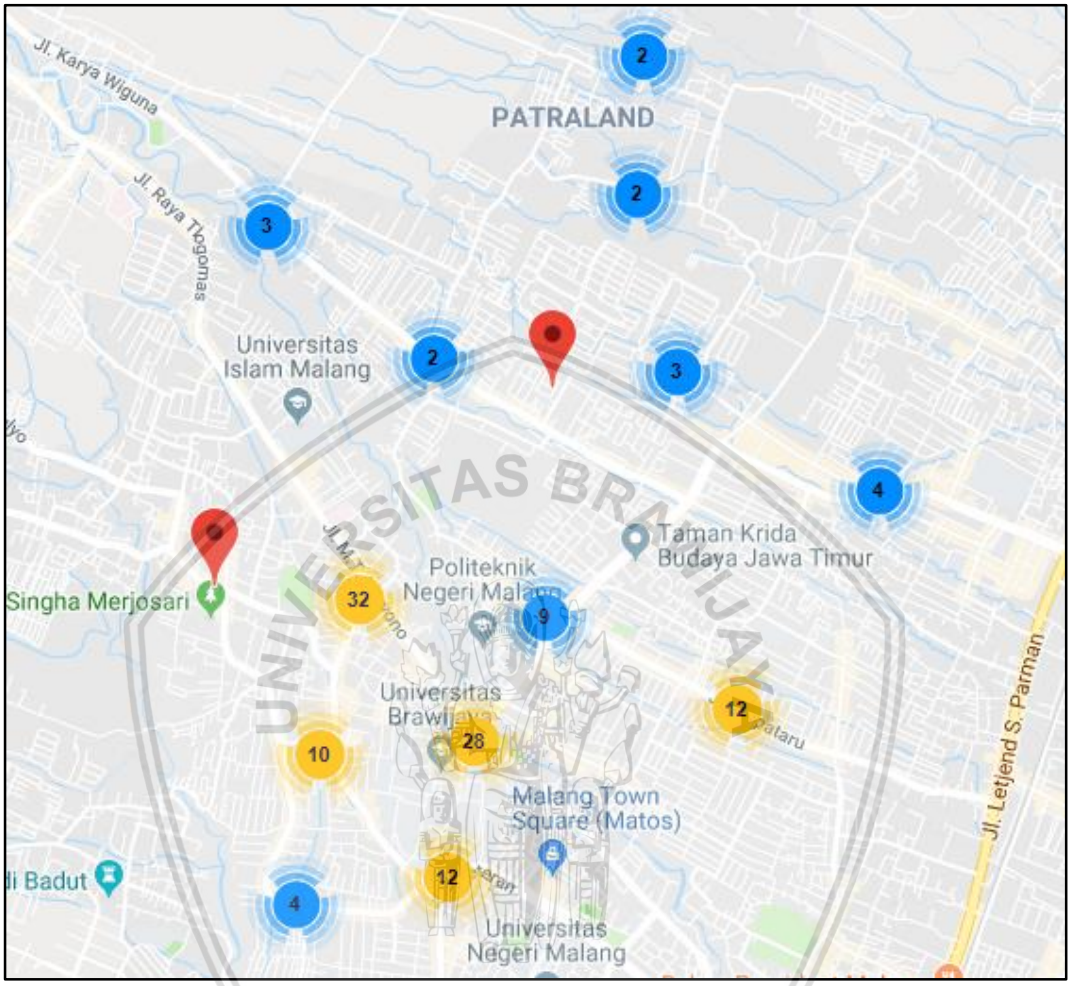
Nomor	*Node*
1	[112.593124,-7.943738]
2	[112.593383001,-7.9438450009999]
3	[112.593583989,-7.943935633]
4	[112.593811,-7.944038]
5	[112.594223,-7.944188]
6	[112.594604,-7.943308]
7	[112.595054,-7.942257]
8	[112.59523,-7.942257]
9	[112.596084,-7.940519]
10	[112.59645,-7.940626]
11	[112.597244001,-7.940948]
12	[112.598274,-7.941356]
13	[112.599411,-7.94157]
14	[112.59974466542,-7.9417464055355]
15	[112.600311,-7.9419570009999]
16	[112.600379,-7.9419350009999]
17	[112.601244363,-7.9415087159999]
18	[112.601364,-7.94157]
19	[112.601684,-7.940562]
20	[112.602073,-7.9406480009999]
21	[112.602844,-7.940176]
22	[112.603363,-7.940347]
23	[112.603813,-7.940497]
24	[112.604202,-7.940583]
25	[112.604263,-7.940369]
26	[112.604324,-7.940304]
27	[112.60527,-7.939360001]
28	[112.605934,-7.93891]
29	[112.60684984882,-7.9384087028929]
30	[112.607032,-7.9383090009999]

31	[112.607864,-7.939489]
32	[112.60791,-7.939575001]
33	[112.6081,-7.939854]
34	[112.608192,-7.940004001]
35	[112.608573,-7.940605]
36	[112.60881,-7.940862001]
37	[112.609993,-7.939961]
38	[112.61042,-7.940562]
39	[112.611,-7.941034]
40	[112.611984,-7.938674]
41	[112.614173,-7.941055]
42	[112.61454,-7.941206]
43	[112.61483356564,-7.9413979811897]
44	[112.616813,-7.942643]
45	[112.617462,-7.943008001]
46	[112.61853,-7.943802]
47	[112.618873,-7.944038]
48	[112.619453,-7.944446]
49	[112.620803,-7.942986001]
50	[112.621002,-7.942858]
51	[112.62104,-7.9428150009999]
52	[112.621902,-7.942021]
53	[112.622093,-7.941849]
54	[112.622741,-7.941270001]
55	[112.623512,-7.940605]
56	[112.624,-7.940197]
57	[112.62445,-7.939832001]
58	[112.62471611073,-7.9396120366424]
59	[112.624969,-7.939403001]
60	[112.625503,-7.938352]
61	[112.625633,-7.938437]
62	[112.625701,-7.9383300009999]
63	[112.625938,-7.9378800009999]

64	[112.626403,-7.937043001]
65	[112.62651,-7.9370000009999]
66	[112.62709,-7.937214001]
67	[112.630462,-7.938888]
68	[112.630523,-7.93891]
69	[112.632202,-7.939382001]
70	[112.632949,-7.939274]
71	[112.633163,-7.93906]
72	[112.63365297136,-7.9388297990417]
73	[112.634323,-7.938824]
74	[112.63533,-7.9391460009999]
75	[112.635551,-7.939124]
76	[112.637138,-7.938588]
77	[112.637626,-7.9374720009999]
78	[112.63797,-7.936785001]
79	[112.638229,-7.936292]
80	[112.638122,-7.93627]
81	[112.6378295124,-7.9361328964362]
82	[112.637802,-7.93612]
83	[112.637542,-7.9359700009999]
84	[112.637268,-7.93582]
85	[112.636962,-7.935669]
86	[112.637481,-7.934725]
87	[112.637611,-7.934425001]
88	[112.63694,-7.934082]
89	[112.636962,-7.934017001]
90	[112.637199,-7.933674]
91	[112.638511,-7.931957]
92	[112.640892,-7.9331800009999]

LAMPIRAN B DATA TITIK DAN WAKTU KEMACETAN

B.1 Persebaran Titik Kemacetan Dari Hasil Survei



Dari gambar persebaran titik kemacetan dari hasil survei (Lampiran B.1), diketahui survei kemacetan tertinggi berada di lampu merah dinoyo sebanyak 32 orang yang menandai daerah tersebut sering terdapat kemacetan dan terendah berada di jalan borobudur dan perempatan merjosari sebanyak 1 orang yang menandai daerah tersebut.

B.2 Verifikasi Oleh Pihak Polsek Lowokwaru

Berikut ini adalah hasil verifikasi dari Polsek Lowokwaru mengenai data titik dan kemacetan. Hasil verifikasi didapatkan dengan cara melakukan wawancara tidak terstruktur pada Polsek Lowokwaru.

Pertanyaan : Dari data survei yang sudah dilakukan (Lampiran B.1), sering terjadi kemacetan pada sekitar lampu merah dinoyo, jembatan suhat, pertigaan itn, uin, jembatan sigura-gura, perempatan merjosari, pizza hut suhat, jalan kalpataru, simpang 5 tunggulwulung, jalan sudimoro, jalan ikan gurami dan jalan K.H Yusuf. Apakah benar daerah tersebut adalah titik-titik kemacetan di Kecamatan Lowokwaru, atau hanya terjadi kepadatan ataupun gangguan jalan seperti perbaikan jalan?

Jawaban : Titik-titik kemacetan hanya terjadi pada lampu merah dinoyo, jembatan suhat, perempatan veteran, sardo dan simpang lima tunggulwulung. Pada titik lainnya hanya terjadi kepadatan, penyebabnya seperti terjadi kemacetan pada jembatan suhat yang cukup panjang, sehingga berdapak kepadatan dibelakangnya yaitu pada pizza hut dan kalpataru , ataupun sedang terjadi perbaikan jalan seperti jembatan sigura-gura.

Pertanyaan : Dari titik-titik kemacetan tersebut, jam-jam berapakah terjadi kemacetan?

Jawaban : Pada lampu merah dinoyo, jembatan suhat dan perempatan veteran pada hari senin sampai jumat, sekitar jam 06.00-07.30, lalu di siang hari sekitar jam 11.30-13.00, lalu jam pulang sekitar jam 16.00-19.00, namun pada hari jumat siang biasanya lancar, pada hari sabtu hanya pada sore sampai malam yaitu sekitar jam 16.00-21.00, pada hari minggu hanya sore sekitar jam 15.30-19.00. Pada simpang lima tunggulwulung hari senin sampai jumat sekitar jam 06.00-07.30, lalu siang hari sekitar jam 11.30-13.00, lalu jam pulang sekitar jam 16.00-19.00, untuk hari jumat siang lancar, pada hari sabtu dan minggu lancar. Pada sardo hari senin sampai jumat pada pagi hari sekitar jam 06.00-07.30 dan sore hari jam 17.00-19.00, untuk hari sabtu sekitar sore sampai malam hari sekitar jam 17.00-21.00.