BAB III
ANALISIS DAN PERANCANGAN SISTEM

III.1. Analisis Sistem

Analisis sistem bertujuan untuk mengidentifikasikan permasalahan-permasalahan yang terdapat pada sistem serta menentukan kebutuhan-kebutuhan dari sistem yang dibangun.

III.2. Analisis Masalah

Salah satu kendala dalam transportasi adalah kemacetan, kemacetan lalu lintas merupakan suatu permasalahan yang seringkali muncul baik pada pengguna transportasi, untuk itu para pengguna transportasi memerlukan solusi rute perjalanan yang terdekat untuk sampai ke tempat tujuan untuk menghindari kemacetan sehingga dapat mengefisienkan waktu, tenaga, serta biaya. Sehingga dari permasalah diatas, diperlukan suatu sistem yang dapat memberikan solusi rute terpendek bagi pengguna transportasi, baik transportasi umum ataupun pribadi berdasarkan data yang ada seperti, rute jalan dan data kemacetan.

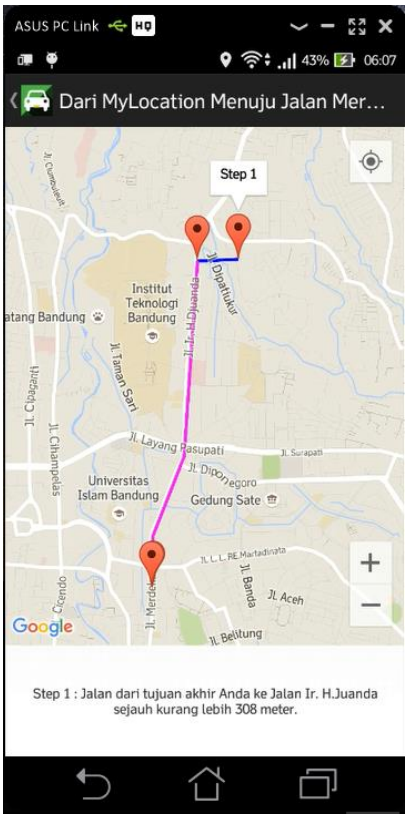
III.3. Analisis Aplikasi Sejenis

Analisis aplikasi sejenis dimaksudkan untuk menganalisis fungsionalitas dan alur dari aplikasi yang telah ada pada domain yang sama dengan perangkat lunak pada penelitian ini. Hal ini bertujuan untuk memilah fungsionalitas dan alur yang akan diadopsi pada sistem yang diteliti berdasarkan manfaat yang sudah dihasilkan dari aplikasi yang dianalisis dan mencari uniqueness dari perangkat lunak yang akan dibangun pada penelitian ini.

Pada penelitian ini terdapat dua aplikasi yang akan diteliti diantaranya adalah Rute Angkot Bandung dan Google Maps dengan pertimbangan kedua aplikasi tersebut berada pada *platform mobile* android dan memiliki jumlah unduhan diatas 500.000. Paramater pembanding yang digunakan mengacu kepada komponen-komponen fungsional.

1. Rute Angkot Bandung

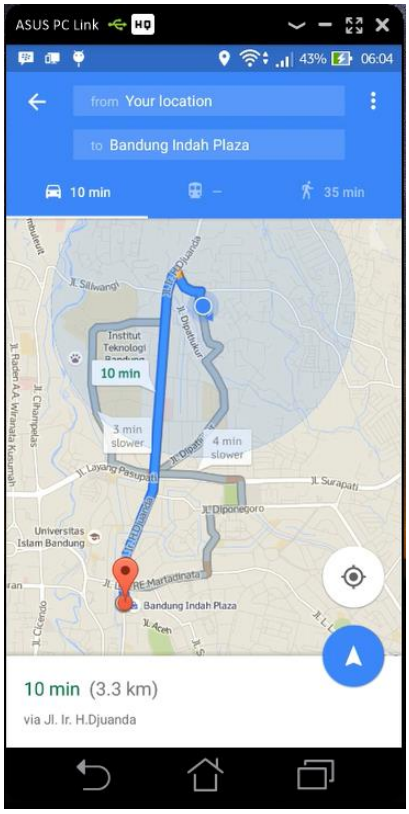
Rute Angkot Bandung adalah aplikasi *mobile* yang dikembangkan dengan berfokus pada permasalahan *urban mobility* koa Bandung yang meneliti permasalahan transportasi angkot, memberikan pencarian rute angkot. Pengguna menggunakan aplikasi ini hanya untuk melakukan pencarian rute angkot. Berikut adalah Gambar III. 1 *Screenshoot* Rute Angkot Bandung.



Gambar III. 1 *Screenshoot* Rute Angkot Bandung

2. Google Maps

Google maps adalah aplikasi *mobile* yang dikembangkan dengan berfokus pada memberikan rute transportasi, *traffic*, dan *timeline* pengguna. Pengguna menggunakan aplikasi ini hanya untuk melakukan pencarian rute menggunakan mode kendaraan pribadi (mobil), kereta api, dan pejalan kaki. Berikut adalah Gambar III. 2 Screenshoot Google Maps.



Gambar III. 2 *Screenshoot* Google Maps

Berdasarkan fungsi pada masing-masing aplikasi tersebut maka dapat dilakukan analisis perbandingan dari kedua aplikasi diatas dengan menggunakan parameter komponen fungsional. Berikut Tabel III. 1 Analisis Perbandingan Aplikasi **Sejenis**.

Tabel III. 1 Analisis Perbandingan Aplikasi Sejenis

Komponen Pembanding	Rute Angkot Bandung	Google Maps
Input	Masukan berupa lokasi tujuan	Masukan berupa lokasi awal dan lokasi tujuan
Proses	Data masukan diproses menjadi fungsionalitas pencarian rute angkot	Data masukan diproses menjadi beberapa fungsionalitas seperti *pencarian rute pejalan kaki, kereta api, pengendara mobil.*
Output	rute angkot dan nama angkot sesuai dengan *input*an lokasi tujuan dari lokasi pengguna, memberikan detail kegiatan apa yang dilakukan untuk mendapatkan angkot, dan didalam angkot.	rute transportasi sesuai dengan masukan tujuan dan asal pengguna, memberikan detail rute pilihan sesuai dengan lokasi tujuan.

Komponen Pembanding	Rute Angkot Bandung	Google Maps
Teknologi	Teknologi google maps api untuk mendukung menampilkan maps get direction dalam pencarain rute	Teknologi google maps api untuk mendukung menampilkan maps get direction dalam pencarain rute
Software	Aplikasi ini menggunakan sistem operasi android dengan minimal version 2.3 Gingerbread.	Aplikasi ini menggunakan sistem operasi android dengan minimal version 2.3 Gingerbread dan Sistem Operasi IOS dengan minimal version 6.0.
Hardware	Aplikasi ini menggunakan beberapa perangkat keras yang sudah terpasang default disetip perangkat android diantaranya: Kamera, GPS, dan A-GPS	Aplikasi ini menggunakan beberapa perangkat keras yang sudah terpasang default disetip perangkat android diantaranya: Kamera, GPS, dan A-GPS
Data	menggunakan penyimpanan internal dan external pada perangkat *mobile*.	menggunakan penyimpanan internal dan external pada perangkat *mobile*.
Komponen Kendali	Aplikasi ini akan menampilkan hasil pencaraian rute jika penggna telah memberikan masukan lokasi tujuan.	Aplikasi ini akan menampilkan hasil pencaraian rute jika penggna telah memberikan masukan lokasi tujuan dan memberikan pilihan jalur apa yang akan di cari.

Dari hasil analisis kedua aplikasi sejenis, aplikasi yang akan dibangun akan memiliki beberapa fungsionalitas diantaranya:

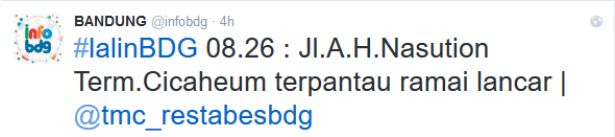
1. Pencarian rute angkot, pejalan kaki dan pengendara mobil menggunakan algoritma dijkstra.
2. Memberikan informasi kondisi jalan
3. Memberikan kooordinat lokasi pengguna

Berdasarkan analisis tersebut juga dapat di*node*kan keunikan perangkat lunak yang dibangun dibandingkan dengan aplikasi sejenis yaitu dukungan kolaborasi pencarian rute, dan didukung dengan pencarian rute terdekat menggunakan algoritma Dijkstra.

III.4. Analisis Data Masukan

Data masukan yang digunakan untuk mendapatkan informasi kemacetan adalah bersumber dari *twitter* dengan menggunakan #LalinBDG. Data *tweet*

tersebut didapat dengan memanfaatkan fitur API (*Aplication Interface*) yang telah disediakan oleh *twitter*. Data yang didapat berupa sebuah kalimat dengan panjang maksimal 140 karakter. Contoh data *tweet* yang didapat dari *twitter* dapat dilihat pada Gambar III. 3.



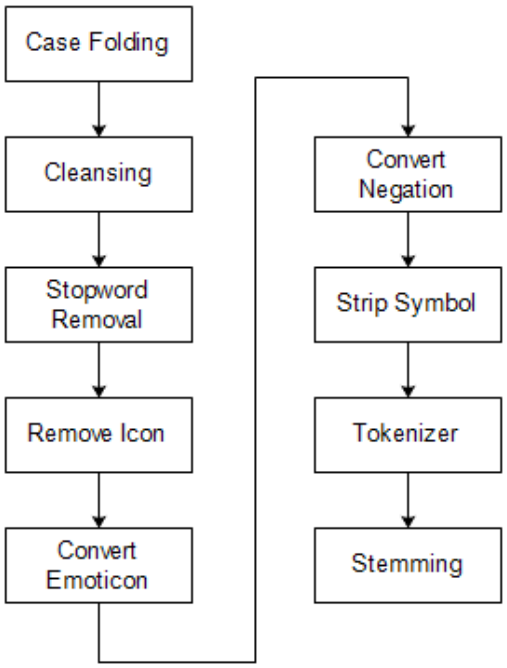
Gambar III. 3 Contoh *tweet* info kemacetan

III.5. Analisis Metode /Algoritma

III.5.1.Analisis *Preprocessing*

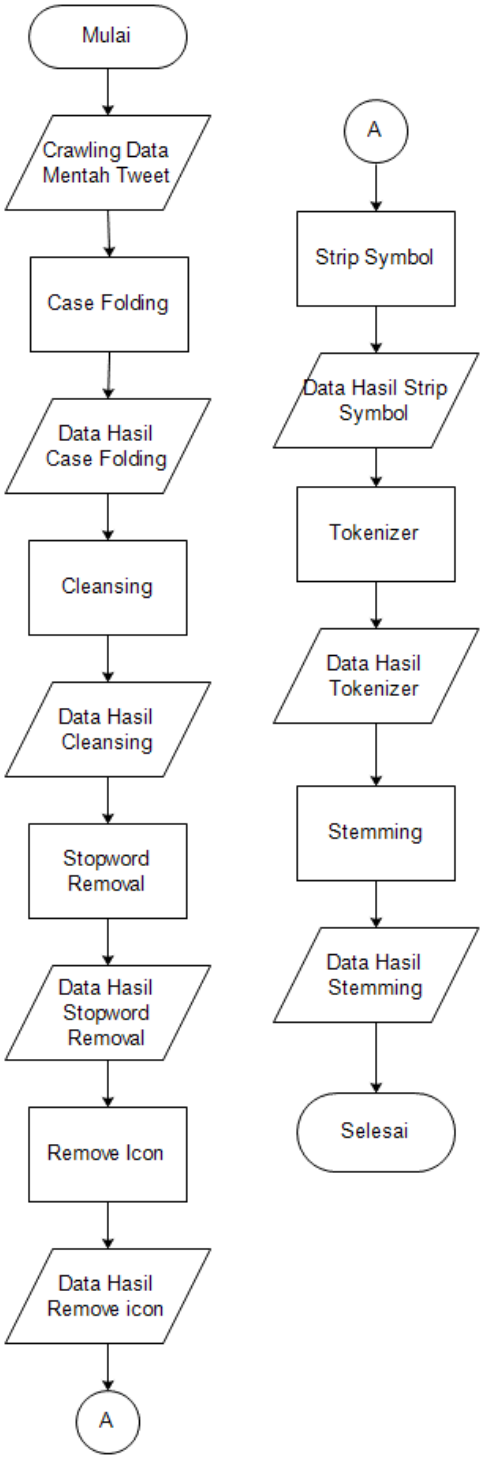
Dalam penenilitian ini data *tweet* dibutuhkan untuk mendapatkan info data kemacetan dari *twitter*, lewat akun @infobdg.

Agar data *tweets* tersebut dapat dimanfaatkan dengan baik untuk mengklasifikasikan kategori, maka dalam penelitian ini diperlukan tahap *preprocessing* terhadap data mentah *(data training).* Dimana pada tahap ini data *tweet* akan dibersihkan dari *noise* seperti link, "RT", *stopword, "@"* dan lain sebagainya. Gambaran proses *preprocessing* dalam penelitian ini dapat dilihat pada Gambar III. 4.



Gambar III. 4 Gambaran proses *preprocessing*

Berikut diagram alir proses *preprocessing* Gambar III. 5



Gambar III. 5 *Flowchart Preprocessing*

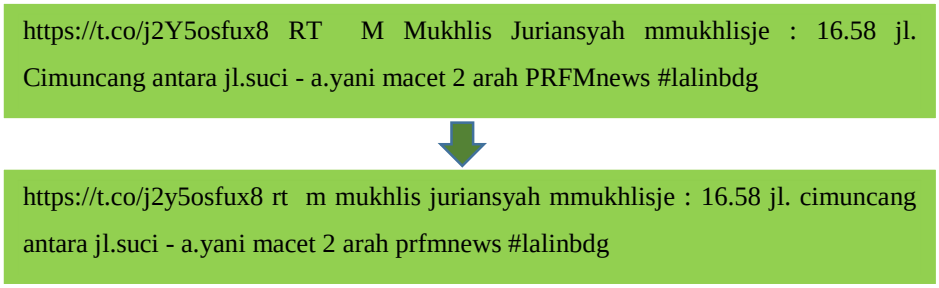
Berikut penjelasan dari proses *preprocessing* secara umum dapat dilihat pada gambar Gambar III. 5:

1. Proses *preprocessing* membutuhkan data *crawling (tweet)*

Data *crawling* telah tersimpan sebelumnya sebelum melakukan proses *preprocessing*.

2. *Case folding*

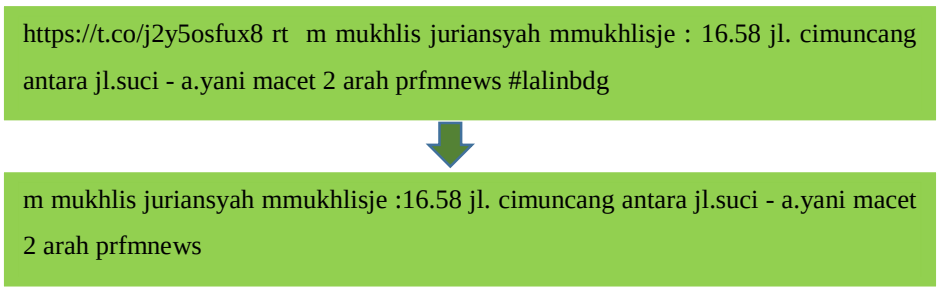
Case folding adalah proses merubah huruf kapital (*uppercase*) menjadi huruf kecil (*lowercase*). Hal ini dilakukan agar semua huruf pada *tweet* menjadi seragam. Gambaran proses dari tahap *case folding* dapat dilihat pada Gambar III. 6.



Gambar III. 6 *Case folding*

3. *Cleansing*

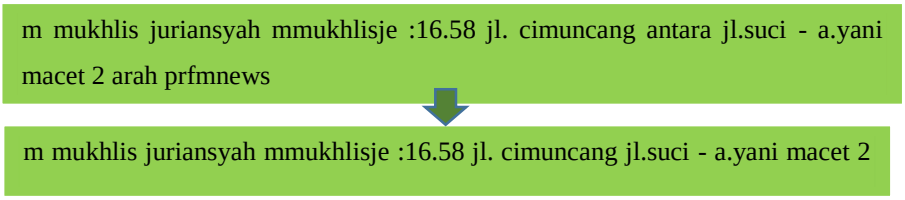
Cleansing merupakan proses menghilangkan *mention*, *hashtag* atau *link* atau membuang komponen yang tidak memiliki pengaruh apapun terhadap *tweet* seperti "@", link, "#" dan RT. Gamabaran proses pada tahap ini dapat dilihat pada Gambar III. 7.



Gambar III. 7 *Cleansing*

4. *Stopword Removal*

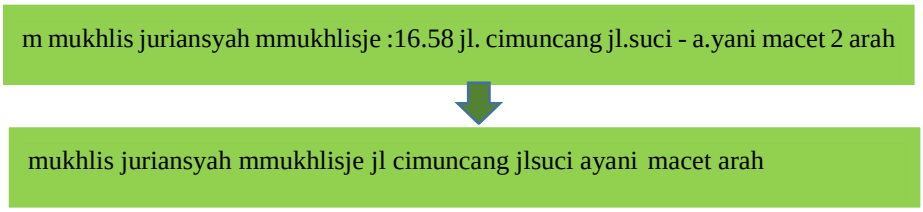
Pada tahap ini *tweet* masih mengandung kata yang dianggap tidak dapat memerikan pengaruh dalam menentukan suatu kategori. Kata-kata tersebut dimasukkan ke dalam daftar *stopword* yang biasa berupa kata ganti orang, kata ganti penghubung, paranomial petunjuk dan lain sebagainya. Jika pada *tweet* yang sudah melalui proses *cleansing* masih terdapat kata yang tercantum pada daftar *stopword* maka kata tersebut dihilangkan. Gambaran proses dari tahap ini dapat dilihat pada Gambar III. 8.



Gambar III. 8 *Stopword Removal*

5. *Remove Icon*

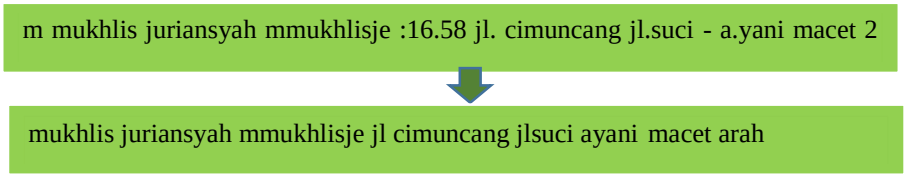
Pada tahap ini *tweet* yang mengandung *icon* akan dihapus karena tidak memberikan pengaruh kepada masukan untuk proses kategori *tweet*. Gambaran proses ini dapat dilihat pada Gambar III. 9.



Gambar III. 9 *Remove Icon*

6. *Strip Symbol*

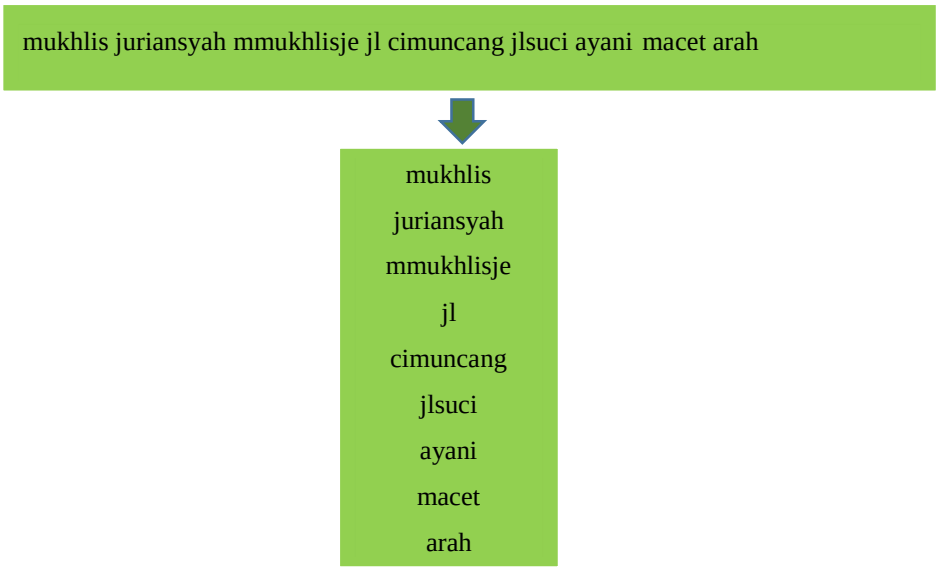
Pada tahap ini simbol yang ada pada *tweet* akan dihapus, dikarenakan tidak dapat memberikan pengaruh untuk proses kategori *tweet*. Gambaran proses ini dapat dilihat pada Gambar III. 10.



Gambar III. 10 *Strip Symbol*

7. *Tokenizing*

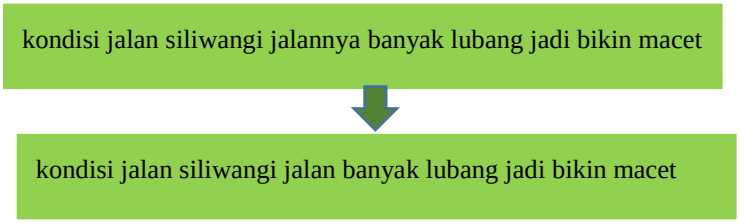
Pada proses *tokenizing* setiap kata pada *tweet* dipisahkan, pada proses ini tahap yang dilakukan adalah memisahkan setiap kata yang dipisahkan oleh spasi. Hal ini dilakukan agar tahap *preprocessing* selanjutnya dapat berjalan. Gambaran proses dari tahap *tokenizing* dapat dilihat pada Gambar III. 11.



Gambar III. 11 *Tokenizing*

8. *Stemming*

Stemming adalah proses mengubah kata berimbuhan kebentuk asalnya (kata dasar). Algoritma yang digunakan untuk proses *stemming* berbahasa Indonesia adalah Nazief dan Andriani. Proses *Stemming* juga termasuk salah satu proses dalam *Information Retrieval*. Gambaran proses dari tahap *stemming* dapat dilihat pada Gambar III. 12.



Gambar III. 12 *Stemming*

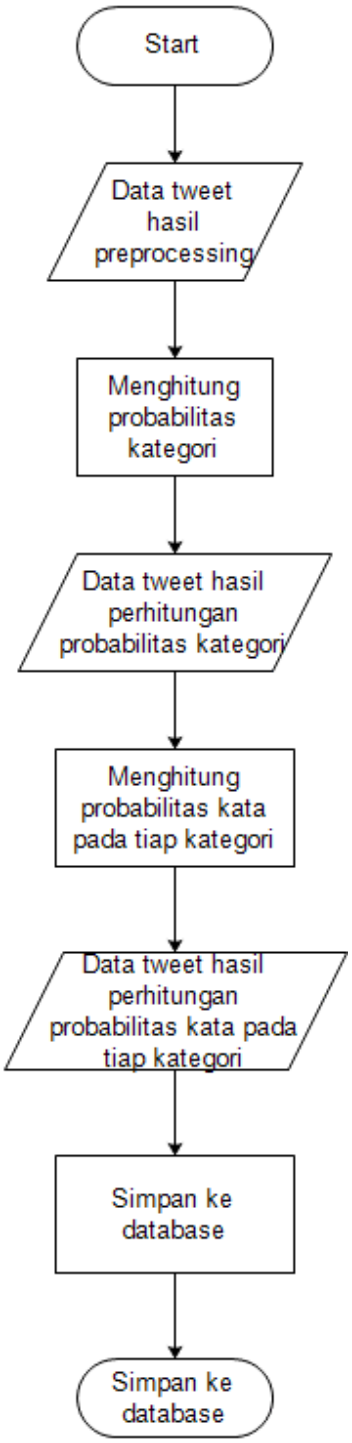
III.5.2.Analisis *Filtering Twitter*

Dalam proses *filtering* ini menggunakan *naïve bayes* ini yang melakukan proses pengklasifikasian *tweet* itu termasuk dalam kategori macet parah, macet, dan lancar. Penentuan standar klasifikasi tersebut dalam penenlitian ini merujuk dari aplikasi *waze*, dikarenakan aplikasi ini telah menjalin kerjasama dengan pemerintah dalam mengatasi kemacetan, yaitu pemerintah DKI Jakarta. [27]

Tahapan-tahapan yang dilakukan untuk membuat kategori menggunakan dari *Naïve bayes Classifier tweet* yang sudah melewati tahap *preprocessing* yang memiliki tahap proses *training* dan proses *testing*, berikut penjabaran proses pada *training* yang dilakukan pada tahap *naïve bayes* untuk menghasilkan probabilitas kata pada setiap kategori :

1. Data *tweet* hasil *preprocessing* sebelumnya digunakan sebagai data masukan untuk melakukan proses *naïve bayes* klasifikasi.
2. Menghitung probabilitas kategori dari hasil *preprocessing*, probabilitas untuk kategori macet, probabilitas kategori macet parah, dan probabilitas kategori lancar. **(**Error! No text of specified style in document.**-5)**
3. Data hasil perhitungan probabilitas kategori ini kemudian akan digunakan untuk menghitung probabilitas kata pada setiap kategori.
4. Data hasil perhitungan probabilitas kata pada setiap kategori ini akan disimpan ke *database*.

Berikut Gambar III. 13 *flowchart* yang menggambarkan proses *training* pada *Naïve bayes Classifier*.

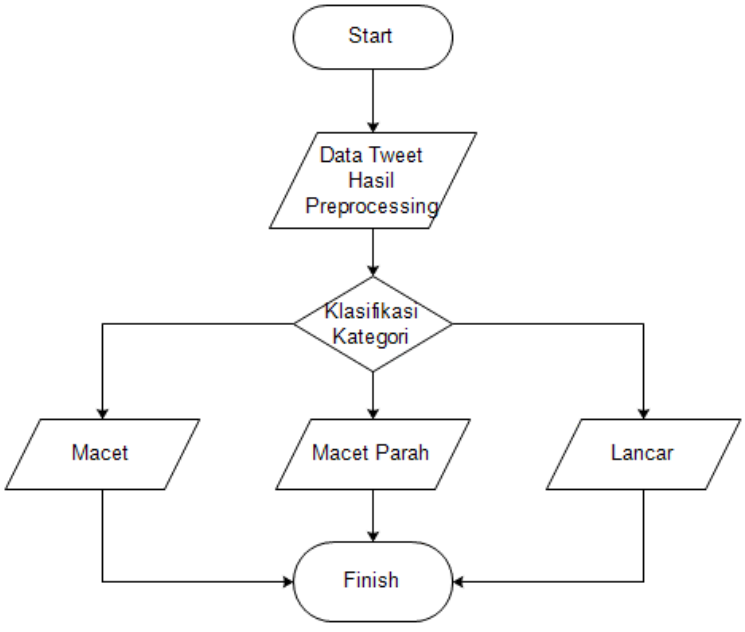


Gambar III. 13 Proses *Training* Naïve Bayes

Berikut penjabaran proses pada tetsting yang dilakukan pada tahap *naïve bayes* untuk menghasilkan kategori sebuah *tweet*:

1. Data *tweet* hasil *preprocessing* sebelumnya digunakan sebagai data masukan untuk melakukan proses *naïve bayes* klasifikasi.
2. Menghitung probabilitas *tweet* pada setiap kategori, probabilitas untuk kategori macet, probabilitas kategori macet parah, dan probabilitas kategori lancar. **(**Error! No text of specified style in document.**-1)**
3. Data hasil perhitungan probabilitas *tweet* pada setiap kategori akan digunakan untuk menghitung perbandingan dengan mencari probabilitas paling besar di setiap kategorinya.
4. Probabilitas *tweet* pada setiap kategori yang akan digunakan sebagai hasil adalah probabilitas yang paling besar.

Berikut Gambar III. 14 *flowchart* yang menggambarkan proses klasifikasi kategori menggunakan *Naïve bayes Classifier*.



Gambar III. 14 Proses *Testing* Naïve Bayes

1. Menghitung Data *Training*

Data training merupakan data perbandingan untuk menentukan kecocokan sebuah tweet termasuk dalam kategori mana, macet parah, macet, atau lancar. Dimana data training ini didefinisikan terlebih dahulu oleh penenliti, dalam penelitian ini kata penentu untuk terkelompok dalam kategori dilihat dari tweet seperi pada Tabel III. 2.

Untuk kategori macet parah dilihat dari *tweet* mengandung kata seperti "macet total", "macet parah", "macet banget".

Untuk kategori macet dilihat dari *tweet* mengandung kata seperti "macet", "padat merayap".

Untuk kategori lancar parah dilihat dari *tweet* mengandung kata seperti "lancar".

Tabel III. 2 *Tweet* Data *Training*

Tweet	Kategori	Fitur
*Tweet*1	macet parah	fly over pasupati arah menuju gasibu macet total hindari guys
*Tweet*2	macet	djuanda terpantau padat merayap cuaca panas sing sabar nya
*Tweet*3	lancar	kawasan jln surapati kearah pusdai sebaliknya lancar
*Tweet*4	lancar	jln sate ramai lancar cuaca cerah

Dibentuk sebuah model probabilistik, dengan mengacu pada persamaan (**II.5**) dan persamaan (**II.6**).

$$P(w_{macet\ total}|c_{macet\ parah}) = \frac{\#words(w_{macet\ total},c_{macet\ parah})+1}{\#words(c_{macet\ parah})+|V|} = \frac{1+1}{10+30} = \frac{2}{40}$$

$$P(w_{macet}|c_{macet}) = \frac{\#words(w_{macet},c_{macet})+1}{\#words(c_{macet})+|V|} = \frac{0+1}{9+30} = \frac{1}{39}$$

$$P(w_{lancar}|c_{lancar}) = \frac{\#words(w_{lancar},c_{lancar})+1}{\#words(c_{lancar})+|V|} = \frac{2+1}{13+30} = \frac{3}{43}$$

Jika dibentuk sebuah tabel, maka hasil dari perhitungan probabilistik setiap kata pada data *training* terdapat pada Tabel III. 3 sampai Tabel III. 7.

Tabel III. 3 Hasil perhitungan probabilitas data *training* (1)

Kategori	$P(c)$	$P(w_{kj}\mid c)$							
		flyover	pasopati	arah	menuju	gasibu	macet	total	hindari
Macet parah	$\frac{1}{4}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{2}{40}$
Macet	$\frac{1}{4}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{39}$
Lancar	$\frac{2}{4}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$

Tabel III. 4 Hasil perhitungan probabilitas data *training* (2)

Kategori	$P(c)$	$P(w_{kj}\mid c)$						
		hindari	guys	djuanda	terpantau	padat	merayap	cuaca
Macet parah	$\frac{1}{4}$	$\frac{2}{40}$	$\frac{2}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$
Macet	$\frac{1}{4}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$
Lancar	$\frac{2}{4}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$

Tabel III. 5 Hasil perhitungan probabilitas data *training* (3)

Kategori	$P(c)$	$P(w_{kj}\mid c)$						
		panas	sing	sabar	nya	kawasan	jln	surapati
Macet parah	$\frac{1}{4}$	$\frac{1}{39}$	$\frac{1}{39}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$
Macet	$\frac{1}{4}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$
Lancar	$\frac{2}{4}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$

Tabel III. 6 Hasil perhitungan probabilitas data *training* (4)

Kategori	$P(c)$	$P(w_{kj}\mid c)$						
		ke	arah	pusdai	sebaliknya	lancar	jln	sate
Macet parah	$\frac{1}{4}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$
Macet	$\frac{1}{4}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{2}{39}$	$\frac{1}{39}$	$\frac{1}{39}$
Lancar	$\frac{2}{4}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{1}{43}$	$\frac{3}{43}$	$\frac{2}{43}$	$\frac{2}{43}$

Tabel III. 7 Hasil perhitungan probabilitas data training (5)

Kategori	$P(c)$	$P(w_{kj}\mid c)$			
		ramai	lancar	cuaca	cerah
Macet parah	$\frac{1}{4}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$	$\frac{1}{40}$
Macet	$\frac{1}{4}$	$\frac{1}{39}$	$\frac{2}{39}$	$\frac{1}{39}$	$\frac{1}{39}$
Lancar	$\frac{2}{4}$	$\frac{2}{43}$	$\frac{3}{43}$	$\frac{2}{43}$	$\frac{2}{43}$

Hasil perhitungan probabilitas tersebut digunakan sebagai model probabilistik yang nantinya digunakan sebagai penentuan kategori data *testing*. Contoh kasus data *testing* yang sudah melalui tahap *preprocessing* adalah sebagai berikut:

Tabel III. 8 Data testing

Tweet	Kategori	Fitur
*Tweet*1	?	menuju djuanda macet nih pohon tumbang depan bca
*Tweet*2	?	lalin perempatan jl otista asia afrika arah tegalega ramai lancar

2. Menghitung Data *Testing*

Data *testing* digunakan untuk menghitung kategori *tweet* yang telah melalui tahap *training.* Berikut adalah pemaparan proses yang terjadi untuk mendapatkan kategori dari *tweet* hasil *training* dengan menggunakan persamaan **(II-1).**

Perhitungan Tweet 1:

$$P(Tweet1|C_{macet\ parah})$$
$$= P(w_{menuju}|C_{macet\ parah}) \times P(w_{djuanda}|C_{macet\ parah}) \times P(w_{macet}|C_{macet\ parah}) \times P(w_{nih}|C_{macet\ parah}) \times P(w_{pohon}|C_{macet\ parah}) \times P(w_{tumbang}|C_{macet\ parah}) \times P(w_{depan}|C_{macet\ parah}) \times P(w_{bca}|C_{macet\ parah})$$
$$= \frac{2}{40} \times \frac{1}{40} \times \frac{2}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40}$$
$$= 0.00000000000061035$$

$$P(Tweet1|C_{macet})$$
$$= P(w_{menuju}|C_{macet\ parah}) \times P(w_{djuanda}|C_{macet\ parah}) \times P(w_{macet}|C_{macet\ parah}) \times P(w_{nih}|C_{macet\ parah}) \times P(w_{pohon}|C_{macet\ parah}) \times P(w_{tumbang}|C_{macet\ parah}) \times P(w_{depan}|C_{macet\ parah}) \times P(w_{bca}|C_{macet\ parah})$$
$$= \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39}$$
$$= 0.00000000000018685$$

$$P(Tweet1|C_{lancar})$$
$$= P(w_{menuju}|C_{macet\ parah}) \times P(w_{djuanda}|C_{macet\ parah}) \times P(w_{macet}|C_{macet\ parah}) \times P(w_{nih}|C_{macet\ parah}) \times P(w_{pohon}|C_{macet\ parah}) \times P(w_{tumbang}|C_{macet\ parah}) \times P(w_{depan}|C_{macet\ parah}) \times P(w_{bca}|C_{macet\ parah})$$
$$= \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43}$$
$$= 0.00000000000008556$$

Tabel III. 9 Nilai probabilitas pada data testing Tweet 1

Kategori	Nilai Probabilitas
Macet Parah	0.00000000000061035
Macet	0.00000000000018685
Lancar	0.00000000000008556

Dapat disimpulkan bahwa *Tweet* 1 termasuk dalam kategori macet parah dikarenakan probabilitasnya lebih besar di bandingkan dengan kategori yang lain dengan nilai probabilitas **0.00000000000061035**.

Perhitungan Tweet 2:

$$P(Tweet2|C_{macet\ parah})$$
$$= P(w_{lalin}|C_{macet\ parah}) \times P(w_{perempatan}|C_{macet\ parah}) \times P(w_{jl}|C_{macet\ parah}) \times P(w_{otista}|C_{macet\ parah}) \times P(w_{asia}|C_{macet\ parah}) \times P(w_{afrika}|C_{macet\ parah}) \times P(w_{arah}|C_{macet\ parah}) \times P(w_{tegalega}|C_{macet\ parah}) \times P(w_{ramai}|C_{macet\ parah}) \times P(w_{lancar}|C_{macet\ parah})$$
$$= \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40} \times \frac{1}{40}$$
$$= 0.00000000000000009537$$

$$P(Tweet2|C_{macet})$$
$$= P(w_{lalin}|C_{macet}) \times P(w_{perempatan}|C_{macet}) \times P(w_{jl}|C_{macet}) \times P(w_{otista}|C_{macet}) \times P(w_{asia}|C_{macet}) \times P(w_{afrika}|C_{macet}) \times P(w_{arah}|C_{macet}) \times P(w_{tegalega}|C_{macet}) \times P(w_{ramai}|C_{macet}) \times P(w_{lancar}|C_{macet})$$
$$= \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{2}{39} \times \frac{1}{39} \times \frac{1}{39} \times \frac{2}{39}$$
$$= 0.00000000000000049138$$

$$P(Tweet2|C_{lancar})$$

$$= P(w_{lalin}|C_{lancar}) \times P(w_{perempatan}|C_{lancar}) \times P(w_{jl}|C_{lancar})$$
$$\times P(w_{otista}|C_{macet\ parah}) \times P(w_{asia}|C_{lancar})$$
$$\times P(w_{afrika}|C_{lancar}) \times P(w_{arah}|C_{lancar}) \times P(w_{tegalega}|C_{lancar})$$
$$\times P(w_{ramai}|C_{lancar}) \times P(w_{lancar}|C_{lancar})$$
$$= \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{1}{43} \times \frac{2}{43} \times \frac{3}{43}$$
$$= 0.00000000000000055526$$

Tabel III. 10 Nilai probabilitas pada data testing Tweet 2

Kategori	Nilai Probabilitas
Macet Parah	0.00000000000000009537
Macet	0.00000000000000049138
Lancar	**0.00000000000000055526**

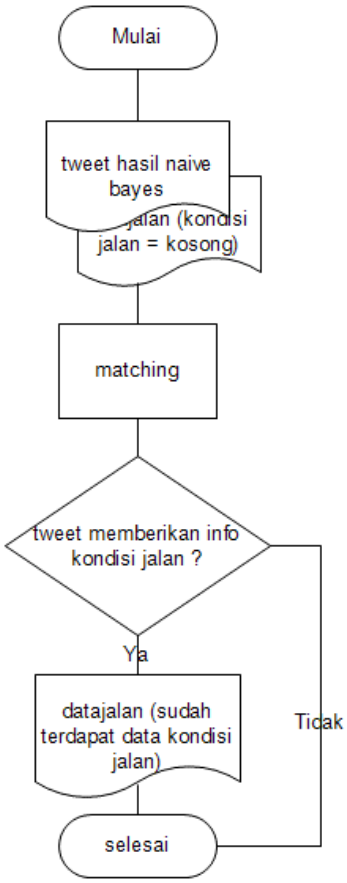
Dapat disimpulkan bahwa *Tweet* 2 termasuk dalam kategori lancar dikarenakan probabilitasnya lebih besar di bandingkan dengan kategori yang lain dengan nilai probabilitas **0.00000000000000055526**.

III.5.3.Analisis *Matching* Nama Jalan Dengan *Tweet*

Dari *tweet* yang telah melewati tahap *Naïve bayes* selanjutnya *tweet* yang tersimpan di dalam *database* sebagai *tweet* hasil proses dari *Naïve bayes* akan digunakan sebagai objek dari proses *matching* nama jalan untuk melihat apakah *tweet* tersebut dapat memberikan informasi kondisi jalan (macet parah, macet dan lancar) atau tidak. Jika dalam *tweet* tersebut memuat kata/*word*/*string* nama jalan, maka *tweet* tersebut dapat memberikan informasi kondisi jalan dan sebaliknya.

Nama jalan telah didefinisikan terlebih dahulu yang tersimpan sebagai *table* nama jalan, kemudian *table* ini memiliki sebuah *field* kondisi jalan yang akan terisi secara otomatis dari hasil proses *matching* nama jalan.

Untuk lebih jelasnya berikut Gambar III. 15 *flowchart* proses *matching* nama jalan dengan *tweet*.



Gambar III. 15 *Flowchart* proses *matching*

Penjelasan dari proses *matching* data jalan dan *tweet* untuk menghasilkan info kemacetan.

1. Data *tweet* yang sudah di kategori (hasil naïve bayes) dan data jalan yang tersimpan pada *database* akan dilkakukan proses pengecekan *string*.
2. Mencari nama jalan yang terkandung dalam *tweet*.
3. Apakah *tweet* mengandung nama jalan, jika mengandung nama jalan *tweet* memberikan informasi kondisi jalan,
4. Jika *tweet* tidak mengandung nama jalan, maka *tweet* tidak memberikan infromasi kondisi jalan.

Dari hasil proses *matching* ini akan memberikan inforasi kondisi jalan yang ada pada setiap *tweet*, apakah kondisi jalan tersebut macet parah, macet, atau lancar.

III.5.4.Analisis Kondisi Lalu Lintas

Kondisi lalu lintas seperti macet, macet parah serta lancar dalam penelitian ini adalah merupakan salah satu bobot untuk perhitungan dijkstra yaitu sebagai bobot kecepatan kendaraan, diamana bobot tiap kategori kondisi lalu lintas mengacu pada penenlitian yang telah dilakukan sebelumnya [28].

Untuk transportasi yang termasuk dalam kategori jalan lokal primer yaitu angkot dan mobil adalah kecepatan rencana paling rendah 20 (dua puluh) km/jam. [9]

Kecepatan kendaraan saat kode berwarna merah dominan adalah sebesar 26,16 km/jam. Saat kode warna kuning dominan, kecepatan default yang digunakan adalah 20,7 km/jam. Saat kode warna merah dominan, kecepatan defaultnya adalah 15,3 km/jam, 12 km/jam saat kode warna hitam dominan. [28]

Dikarenakan dalam penenlitian ini yang di gunakan adalah 3 (tiga) kategori maka dapat disimpulkan bahwa, untuk kondisi jalan macet parah maka kecepatan kendaraan adalah 12 km/jam = 200 m/menit, macet 15 km/jam =250 m/menit, lancar 20 km/jam 333,33 m/menit.

III.5.5. Analisis Proses Konversi Tweet, Nama Jalan, dan Kondisi Jalan

Analisis proses konversi tweet, nama jalan, dan kondisi jalan adalah merupakan suatu proses untuk mendapatkan perubahan kondisi jalan jika terdapat masukan nama jalan memiliki kondisi macet parah, macet atau lancar.

Dimana proses ini akan melakukan perhitungan kecepatan kendaraan dan jarak tempuh serta waktu tempuh. Berikut adalah rumus untuk menghitung waktu tempuh kendaraan : [29]

$$v = \frac{s}{t}$$

Dimana : v = kecepatan (km/jam)

s = jarak tempuh (km, m)

t = waktu tempuh (jam ,s)

untuk waktu dan jarak tempuh normal diambil dari google maps, berikut proses perhitungan untuk mendapatkan waktu tempuh, karena yang menjadi bobot sebagai masukan untuk mendapatkan kondisi jalan adalah waktu tempuh :

$$t = \frac{s}{v}$$

Diketahui :

Tabel III. 11 Jarak tempuh jalan

Nama Jalan	Jarak (km)
Jl. Ir. H. Djuanda	0.24
	1.1
	0.45
	0.19
	0.8
Jalan Surapati	0.35
	0.89
	0.26
	0.28
	1.8
Jalan Tamansari	0.16
	1.6
Jalan P.H.H Mustapa	2.4

Tabel III. 12 Kecepatan kondisi jalan

Kondisi jalan	Kecepatan (km/jam)
Macet parah	12
Macet	15
Lancar	20

Jika kondisi macet parah

$$t = \frac{s}{v}$$

$$t = \frac{0.24}{12}$$

$$t = 1.2\ menit$$

Jika kondisi macet

$$t = \frac{s}{v}$$
$$t = \frac{0.24}{15}$$
$$t = 0.96\ menit$$

Jika kondisi lancar

$$t = \frac{s}{v}$$
$$t = \frac{0.24}{20}$$
$$t = 0.72\ menit$$

III.5.6.Analisis Algoritma Dijkstra

Dalam penelitian ini ada beberapa elemen yang digunakan dalam penerapan Algoritma Dijkstra, antara lain adalah sebagai berikut:

1. Himpunan kandidat, C

   Himpunan ini berisi elemen-elemen yang memiliki peluang untuk membentuk solusi. Pada solusi pencarian rute terpendek himpunan kandidat ini adalah himpunan *node* pada lintasan tersebut, seperti koordinat lokasi pengguna, koordinat lokasi tujuan.
2. Himpunan solusi, S

   Himpunan ini berisi solusi dari permasalahan yang diselsaikan dan elemennya terdiri dari elemen dalam kandidat namun tidak semuanya atau dengan kata lain himpunan solusi ini adalah bagian dari himpunan kandidat itu sendiri.
3. Fungsi seleksi

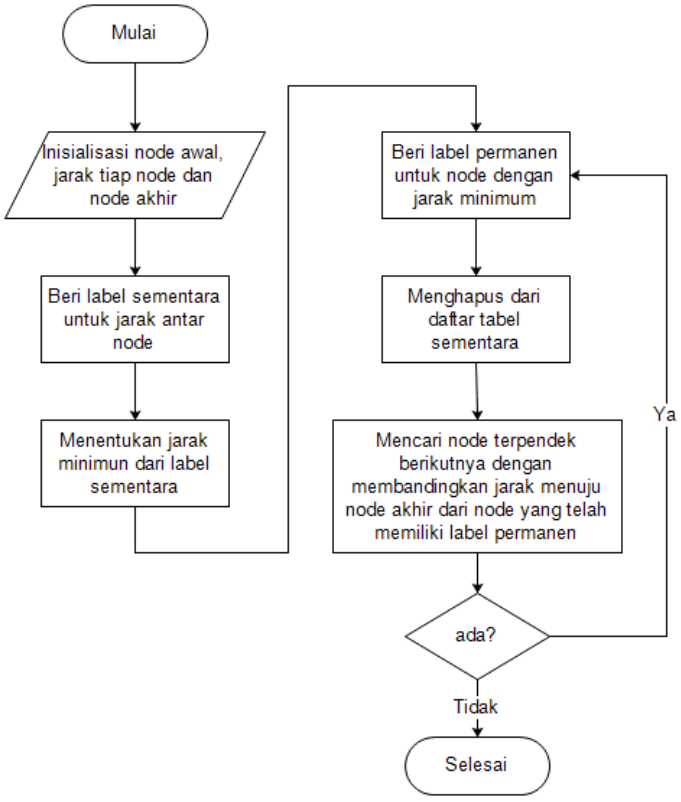
   Fungsi yang akan memilih setiap kandidat yang akan memungkinkan akan menghasilkan solusi optimal pada setiap langkahnya.

4. Fungsi kelayakan

   Fungsi keleyakan akan memeriksa apakah suatu kandidat yangterpilih melanggar atau tidak. Apabila kandidat melanggar maka kandidat tidak akan dimasukkan kedalam himpunan solusi.

5. Fungsi Objektif

   Fungsi ini akan memaksimalkan atau meminimalkan nilai solusi. Tujuannya adalah memilih satu solusi terbaik dari masing-masing anggota himpunan solusi.



Gambar III. 16 *Flowchart* Algoritma Dijkstra

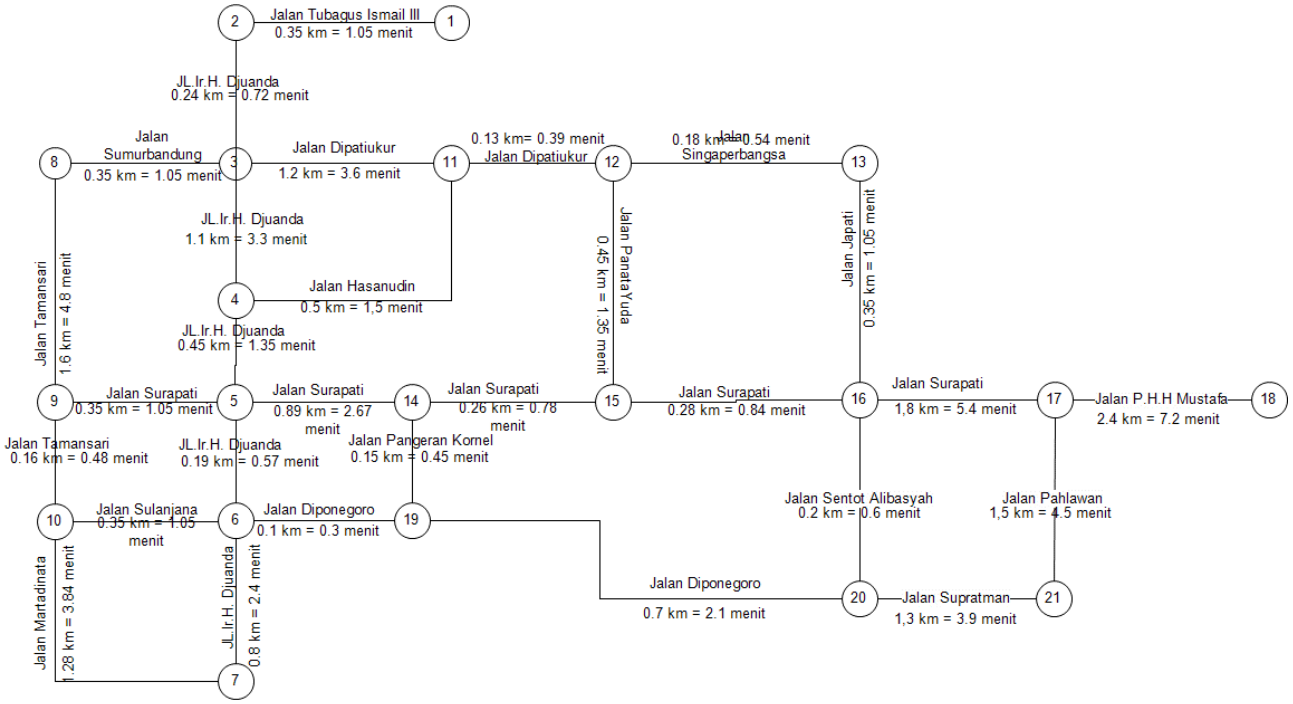
Secara umum langkah-langkah algoritma dijkstra dalam penelitian ini ditulis sebagai berikut jika seandainya ingin mencari rute terdekat dari *node* a :

1. Tetapkan jarak semua *node* terhadap *node* a, yaitu *infinity* atau tak hingga untuk *node* yang lain dan 0 untuk *node* a.
2. Tandai semua *node* dengan status belum dikunjungi. Jadikan *node* awal *node* awal sebagai *node* terkini.

3. Untuk *node* terkini, hitung jarak semua tetangga *node* ini dengan menghitung jarak (dari *node* awal).
4. Setelah selesai mnegecek semua tetangga dari *node* terkini di tandai dengan status sudah dikunjungi.
5. Setelah semua *node* dikunjungi maka akan didapati jarak minimum semua *node* terhadap *node* a.

1. Analisis Algoritma Dijkstra untuk mobil

Proses pencarian rute terdekat untuk angkutan seperti mobil dalam penelitian ini adalah dengan memperhitungkan waktu, dimana contoh kasus yang pertama jika kondisi jalan lancar. Berikut adalah Gambar III. 17 Rute Kondisi Jalan Lancar.



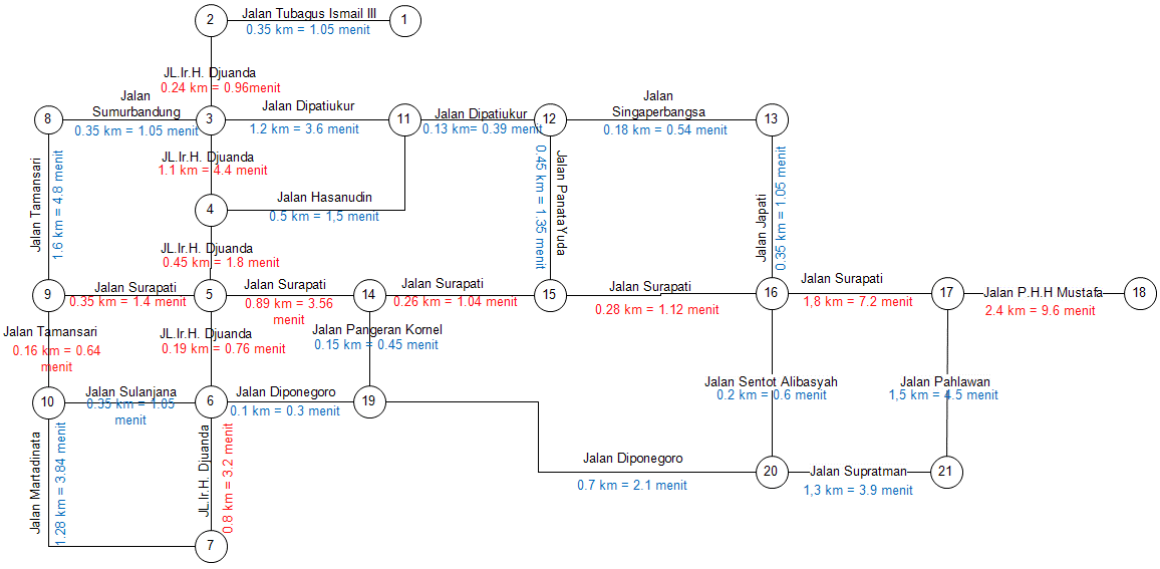
Gambar III. 17 Rute Kondisi Jalan Lancar

Berikut **Tabel III. 13** kondisi ketika jalan mendapatkan nilai macet.

Tabel III. 13 Tabel Perubahan Kondisi Jalan Dalam Satuan Menit

Nama Jalan	Kondisi Lancar (menit)	Kondisi Macet (menit)
Jl. Ir. H. Djuanda	0.72	0.96
	3.3	4.4
	1.35	1.8
	0.57	0.76
	2.4	3.2
Jalan Surapati	1.05	1.4
	2.67	3.56
	0.78	1.04
	0.84	1.12
	5.4	7.2
Jalan Tamansari	0.48	0.64
Jalan P.H.H Mustapa	7.2	9.6

Berikut adalah **Gambar III. 18 Rute Kondisi Jalan Macet**.



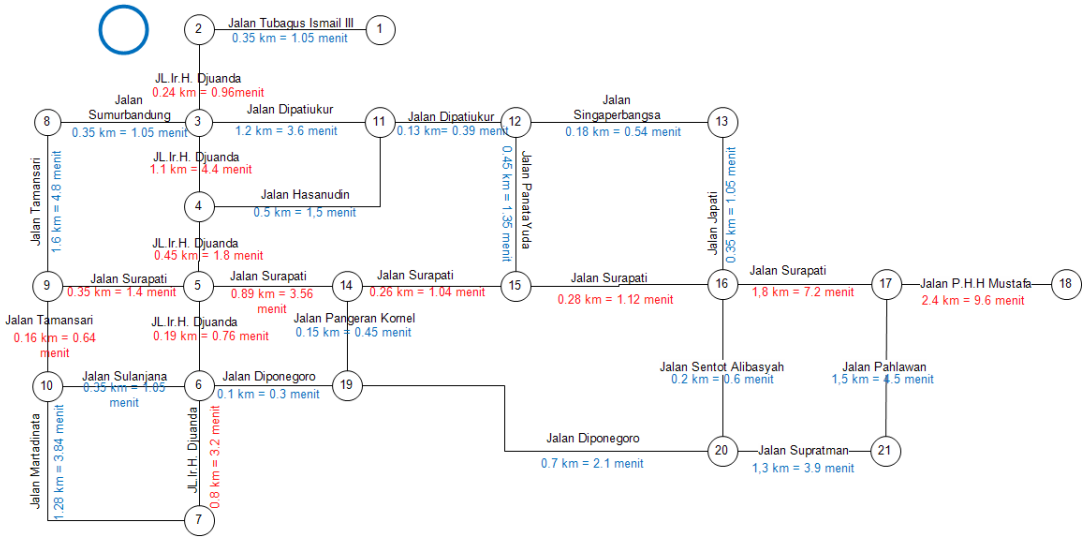
Gambar III. 18 Rute Kondisi Jalan Macet

Penyelesaian per langkah di ilustrasikan dengan penjelasan berikut ini:

{S} adalah himpunan *node* yang jarak terdekatnya telah diketahui C[K] adalah kapasitas busur yang menunjukkan waktu tempuh pada *node* tertentu dari *node* saat ini dimana harga C[K] = 0, menunjukkan *node* saat ini, C[K] = ∞, menunjukkan tidak ada busur yang terhubungkan secara langsung dari *node* saat ini. Untuk

mendapatkan rute terdekat dari *node* 2 ke *node* 19 dilakukan langkah-langkah sebagai berikut:

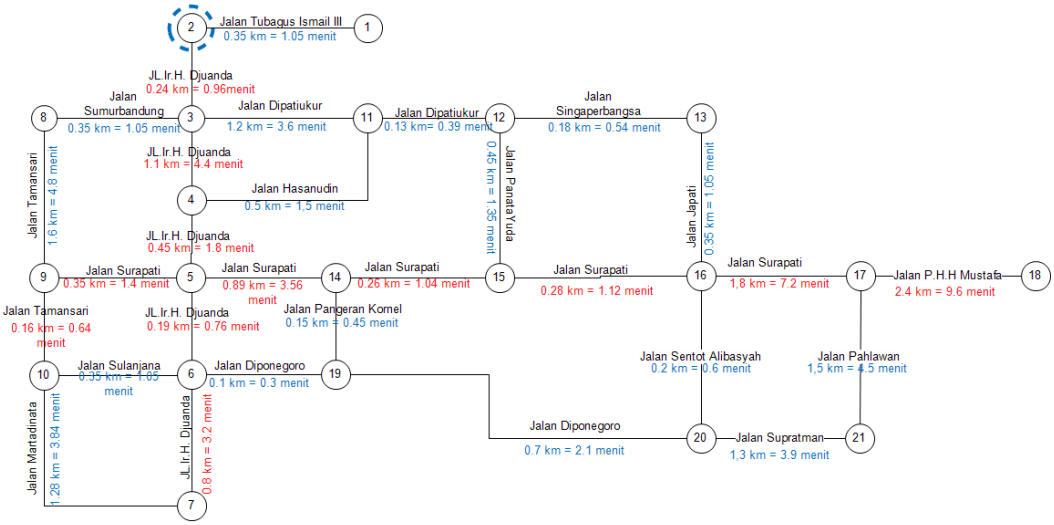
Langkah 0 – Tahap inisisasi dapat dilihat pada Gambar III. 19 Tahap inisiasi dijkstra berikut.



Gambar III. 19 Tahap inisiasi dijkstra

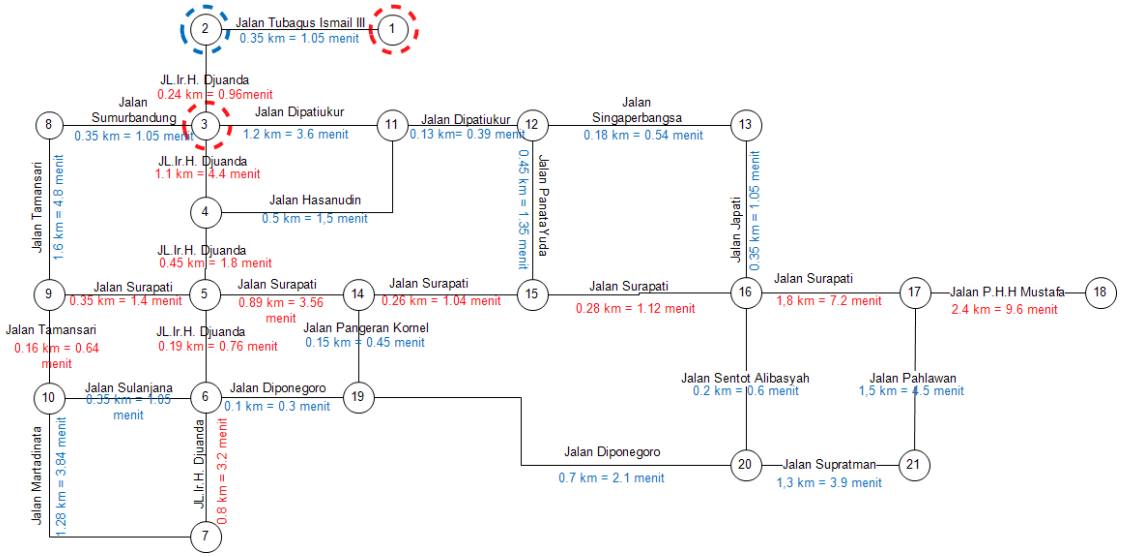
{S} = 0

Langkah 1 – memasukkan *node* K yang dipilih sebagai *node* sumber dan memasukkannya ke dalam himpunan *node* yang diperhitungkan {S} = 2, lihat Gambar III. 20 Tahap identifikasi *node* sumber (awal).



Gambar III. 20 Tahap identifikasi *node* sumber (awal)

Langkah 2 – memperbarui C[K] untuk semua *node* terdekat dari *node* 2 yang tidak ada dalam {S} lihat Gambar III. 21 Tahap ke-3 dijkstra.

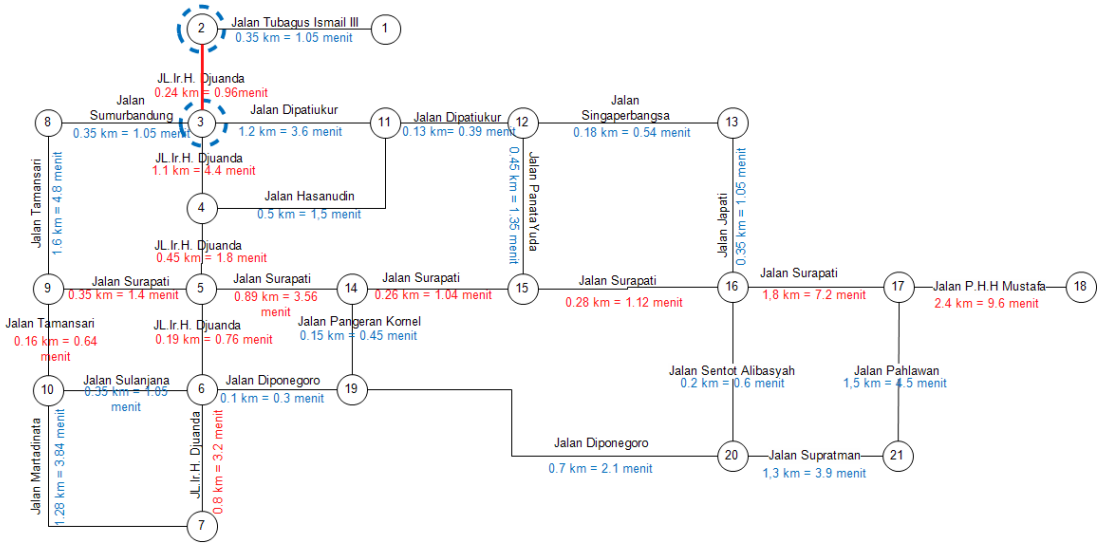


Gambar III. 21 Tahap ke-3 dijkstra

C[2] = min (∞,([2] + (2,1)) = min(∞, 0+1,05) = 1,05

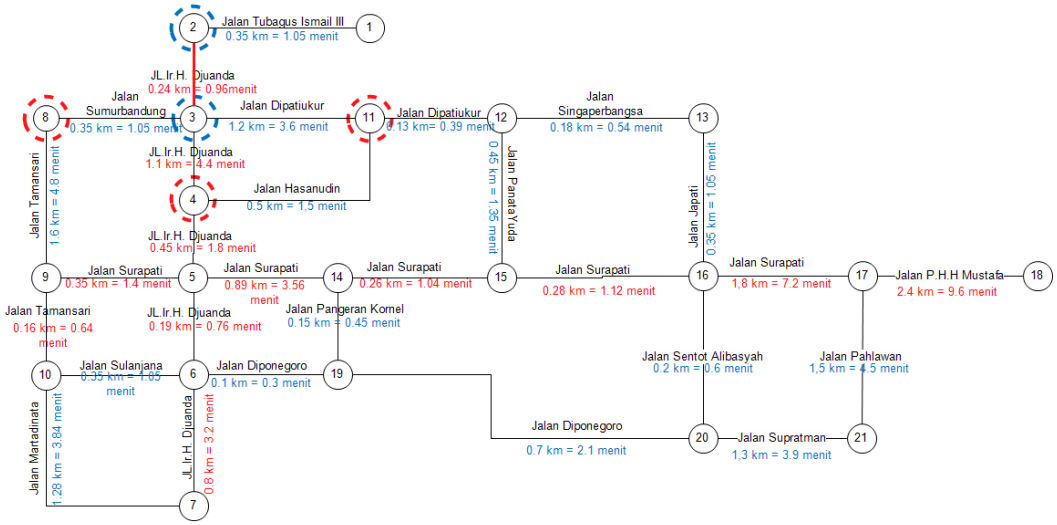
C[3] = min (∞,([2] + (2,3)) = min(∞, 0+0.96) = 0.96

Langkah 3 – Memilih *node* K terpilih dengan jarak terdekat dari *node* A dan ditambah ke {S} = 2, 3, dari perthitungan sebelumnya didapat *node* 3 sebagai rute minimum seperti Gambar III. 22 Tahap ke-4 dijkstra.



Gambar III. 22 Tahap ke-4 dijkstra

Langkah 4 – memperbarui C[K] untuk semua *node* terdekat dari *node* 3 yang tidak ada dalam {S}, Simpan *node* 3 sebagai *node* awal selanjutnya



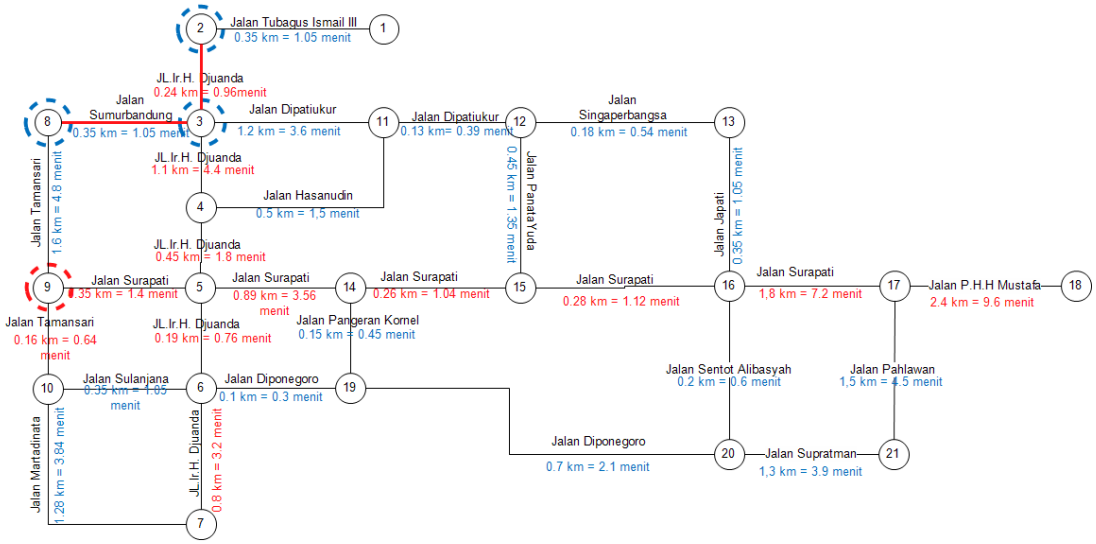
Gambar III. 23 Tahap ke-5 dijkstra

Langkah – 5 memilih *node* K terpilih dengan jarak terdekat dari *node* 3 dan di tambah ke {S} = 2,3,8. *Node* awal selanjutnya adalah 8.

C[8] = 1.05, C[3] = 0.96 , (3,8) = 2.01

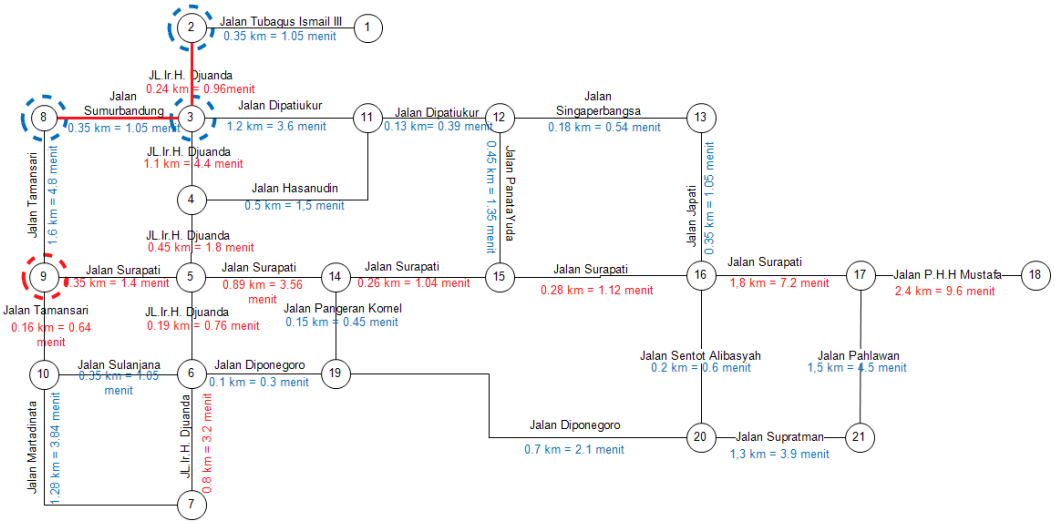
C[4] = 4.4, C[3] = 0.96, (3,4) = 5.36

C[11] =3.6, C[3] = 0.96, (3,11) = 4.56



Gambar III. 24 Tahap ke-6 dijkstra

Langkah ke - 6 Memperbarui C[K] untuk semua *node* terdekat dengan *node* 8, dan *node* 9 merupakan *node* satu-satunya yang terhubung dengan *node* 8, maka dipilih 9 sebagai *node* selanjutnya dan simpan *node* 9 ke dalam {S} = 2,3,8,9.

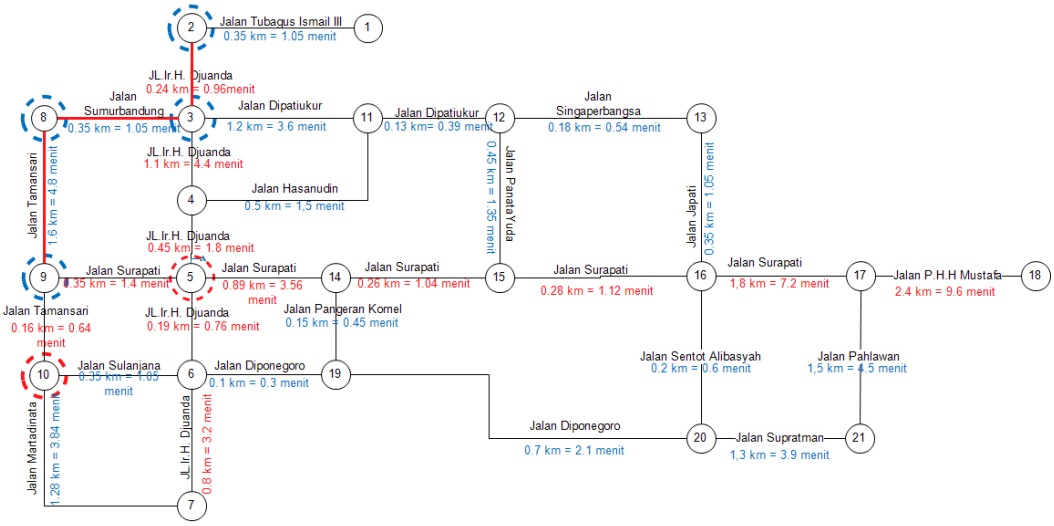


Gambar III. 25 Tahap ke-7 dijkstra

Langkah ke - 7 Memperbarui C[K] untuk semua *node* terdekat dengan *node* 9

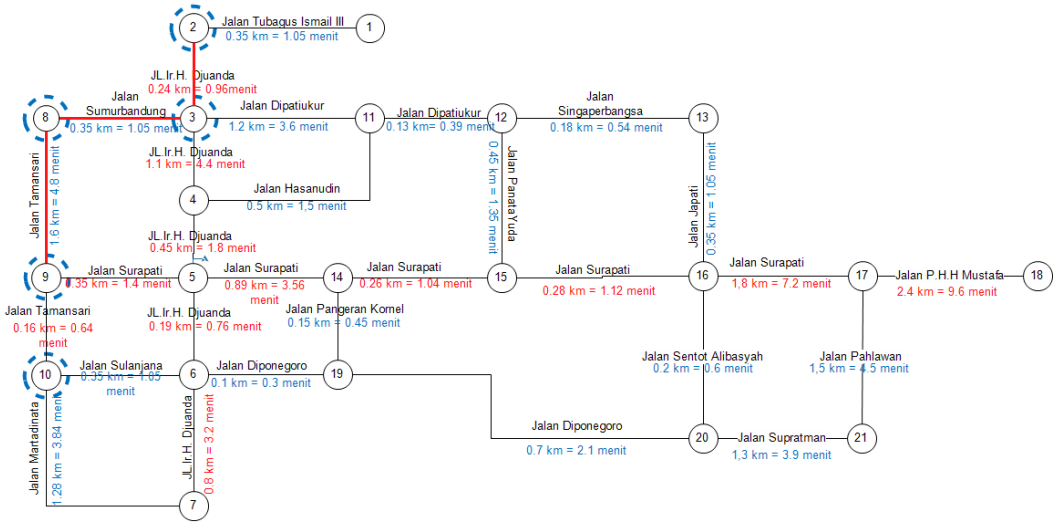
C[5] = 1.4, C[9] = 6.81 , (9,5) = 8.21

C[10] = 0.64, C[9] = 6.81, (9,10) = 7.45



Gambar III. 26 Tahap ke-8 dijkstra

Langkah ke - 8 Memilih *node* terdekat dengan *node* 9, dan terpilih *node* 10 adalah yang terdekat dengan *node* 9, simpan 10 sebagai *node* awal selanjutnya {S} = 2,3,8,9,10.

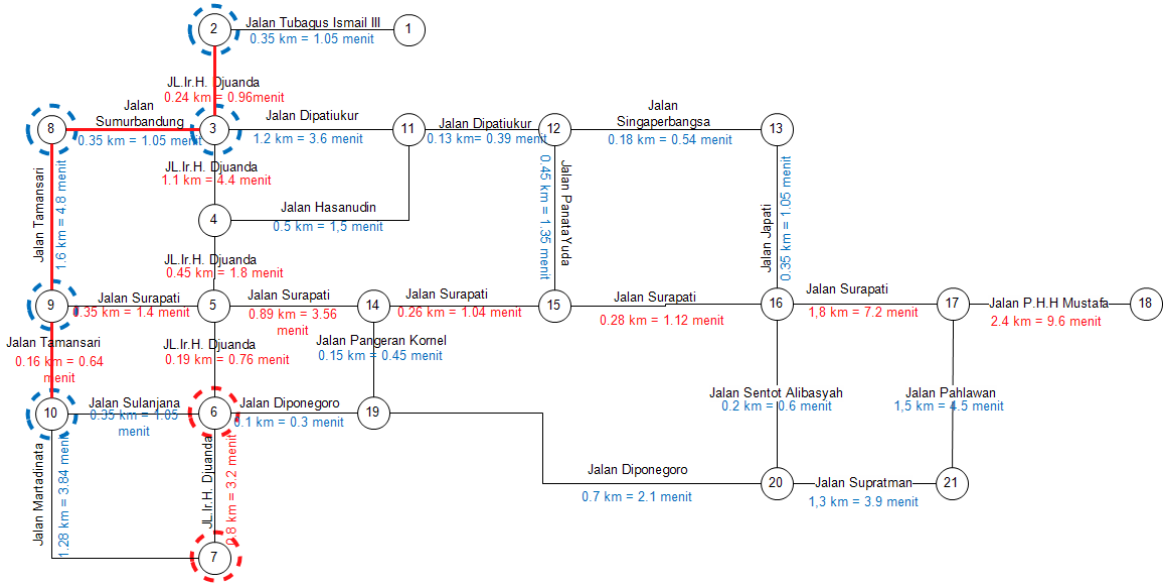


Gambar III. 27 Tahap ke-9 dijkstra

Langkah ke - 9 Memperbarui C[K] untuk semua *node* terdekat dengan *node* 10

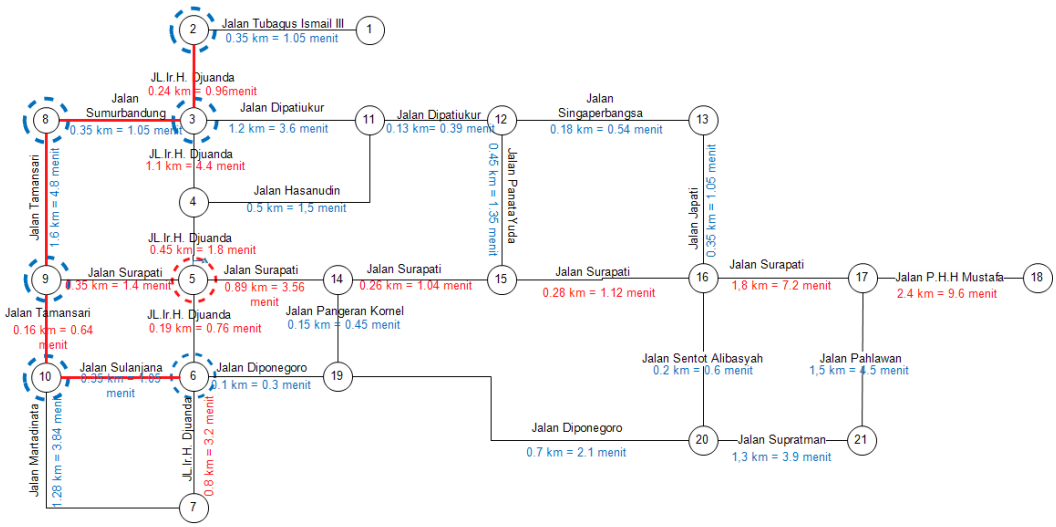
C[6] = 1.05, C[10] = 7.45 , (10,6) = 8.5

C[7] = 3.84, C[10] = 7.45, (10,6) = 11.29



Gambar III. 28 Tahap ke-10 dijkstra

Langkah ke - 10 Memilih *node* terdekat dengan *node* 10, dan terpilih *node* 6 adalah yang terdekat dengan *node* 10, simpan 6 sebagai *node* awal selanjutnya{S} = 2, 3, 8, 9, 10, 6.

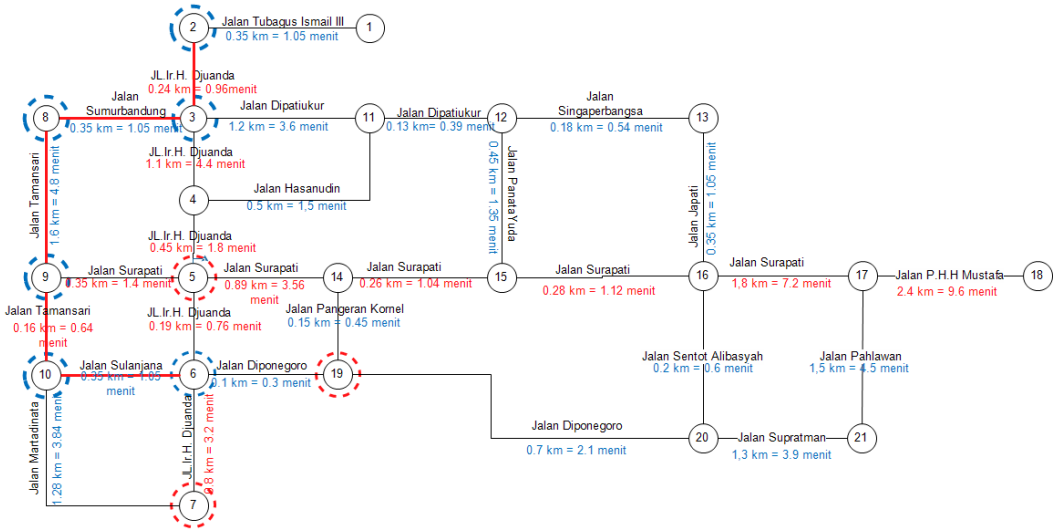


Gambar III. 29 Tahap ke-11 dijkstra

Langkah ke - 11 Memperbarui C[K] untuk semua *node* terdekat dengan *node* 6 yang terpilih adalah *node* 7 dan 19

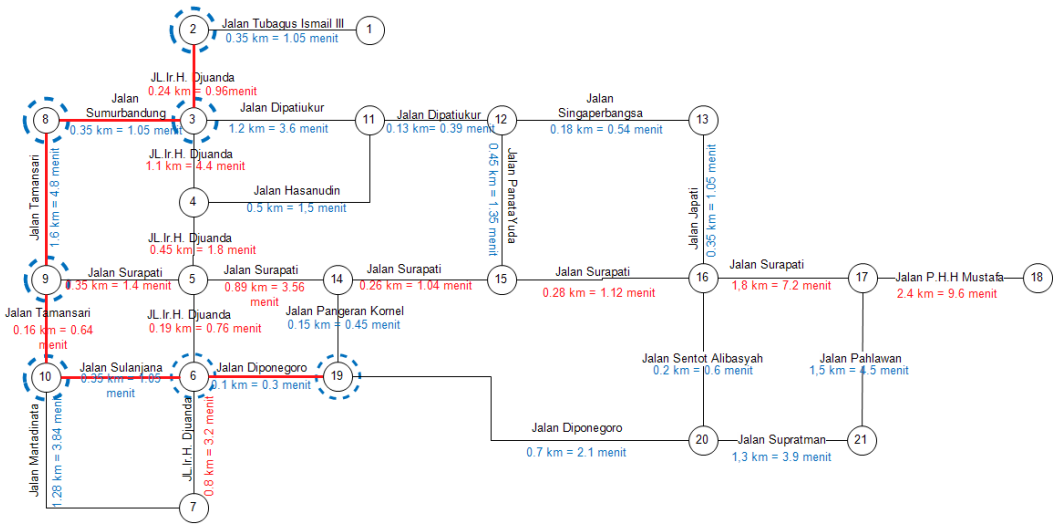
C[7] = 3.2, C[6] = 8.5 , (6,7) = 11.7

C[19] = 0.3, C[6] = 8.5, (6,19) = 8.8



Gambar III. 30 Tahap ke-12 dijkstra

Langkah ke - 12 Memilih *node* terdekat dengan *node* 6, dan terpilih *node* 19 adalah yang terdekat dengan *node* 6, simpan 19 sebagai *node* awal selanjutnya {S}. dikarenakan *node* akhir sudah ditemukan maka rute terdekat adalah {S} = 2, 3, 8, 9, 10, 6, 19.



Gambar III. 31 Tahap ke-13 dijkstra

Rute terpendek yang dapat dilewati kendaraan mobil dari Jalan Tubagus Ismail menuju Jalan Merdeka adalah melewati rute :

1. Jalan Ir.H. Djuanda
2. Jalan Sumur Bandung
3. Jalan Tamansari
4. Jalan Sulanjana
5. Jalan Diponegoro

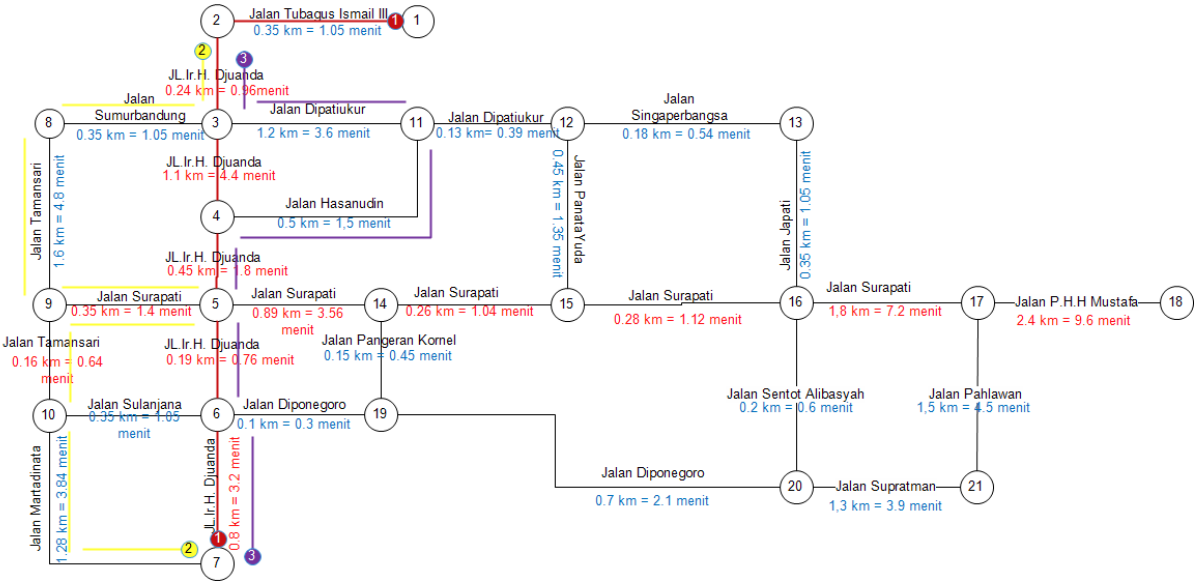
Dengan estimasi waktu 8 menit 8 detik.

2. Analisis Algoritma Dijkstra untuk Angkot

Proses pencarian rute terdekat untuk angkutan angkot dalam penelitian ini adalah dengan memperhitungkan waktu tempuh perjalan, dimana contoh kasus yang pertama jika kondisi jalan macet diamana untuk waktu tempuh rute pada setai jalan terlihat pada Tabel III. 13 Tabel Perubahan Kondisi Jalan Dalam Satuan

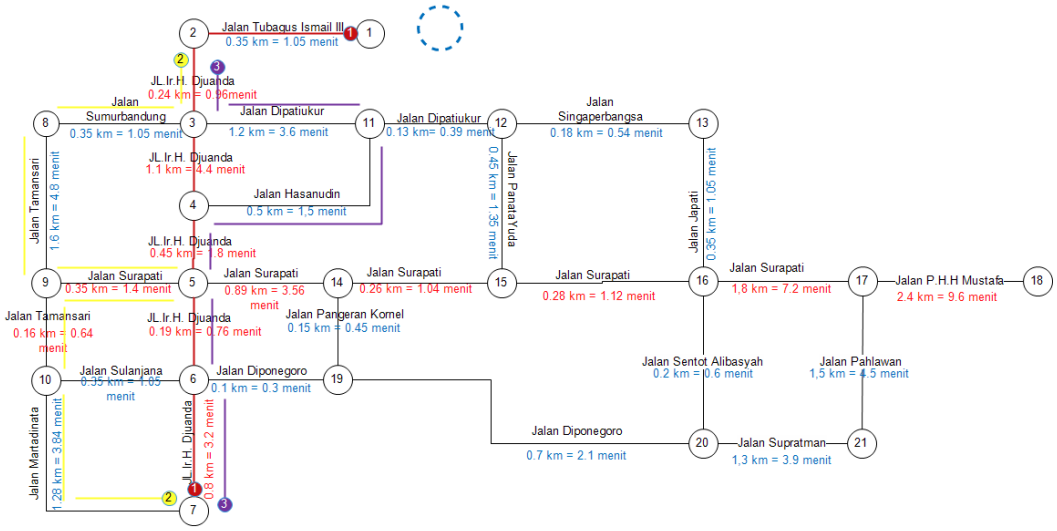
Menit. Berikut adalah Gambar III. 32 Trayek Angkot dimana dalam gambar terdapat 3 angkot dari *node* 1 menuju *node* 7 berikut jalur angkot yang dimaksud :

1. Angkot 1 (*node* merah) : Jalan Tubagus Ismail III – Jl. Ir. H. Djuanda
2. Angkot 2 (*node* kuning) : Jalan Tubagus Ismail III – Jl. Ir. H. Djuanda – Jl. Sumur Bandung – Jl. Tamansari – Jl. Surapati – Jl. Tamansari – Jl. Martadinata
3. Angkot 3 (*node* ungu) : Jalan Tubagus Ismail III - Jl. Ir. H. Djuanda – Jl. Dipati Ukur – Jl. Hasanudin - Jl. Ir. H. Djuanda



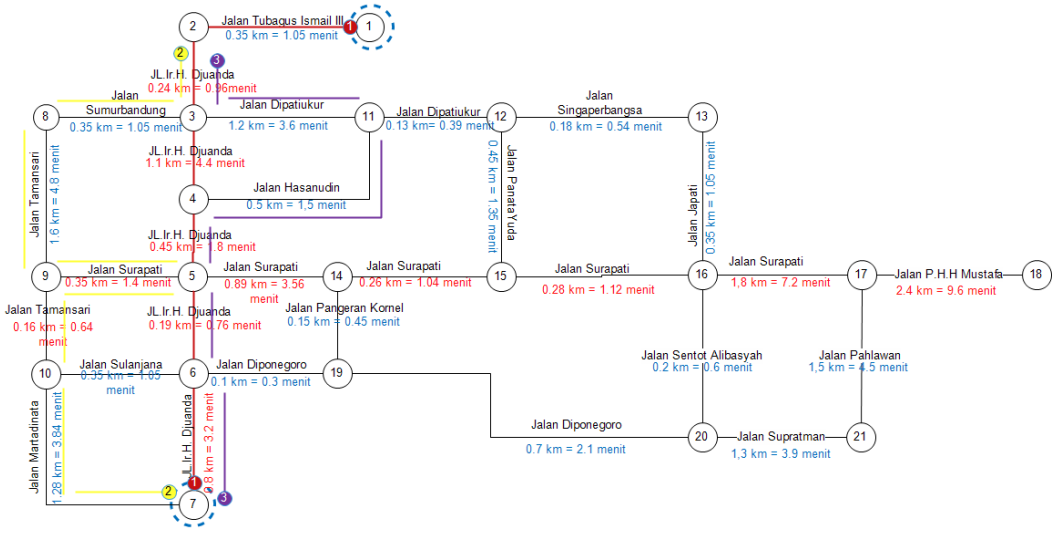
Gambar III. 32 Trayek Angkot

Langkah 0 – Tahap inisisasi dapat dilihat pada Gambar III. 33.



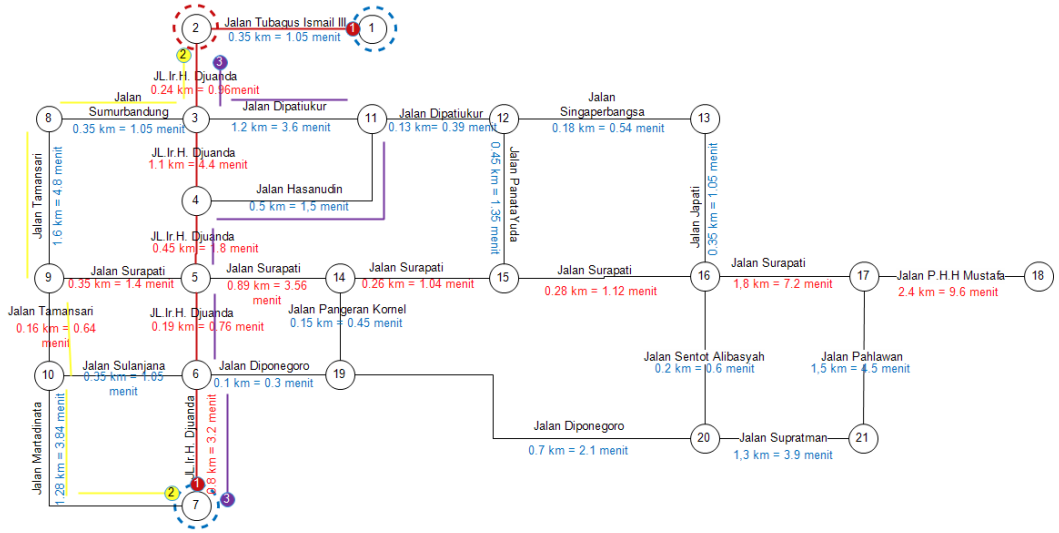
Gambar III. 33 Tahap Insiasi Rute Angkot

Langkah 1 – memasukkan *node* K yang dipilih sebagai *node* sumber dan memasukkannya ke dalam himpunan *node* yang diperhitungkan {S} = 1.



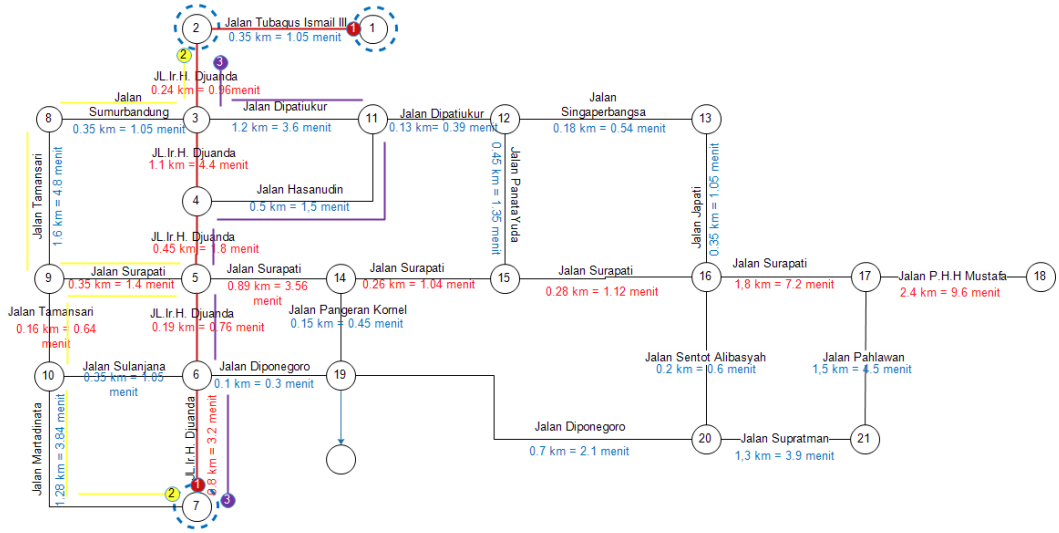
Gambar III. 34 Identifikasi *node* sumber dan tujuan

Langkah 2 – memperbarui C[K] untuk semua *node* terdekat dari *node* 1 yang tidak ada dalam {S}, Simpan *node* 2 sebagai *node* awal selanjutnya



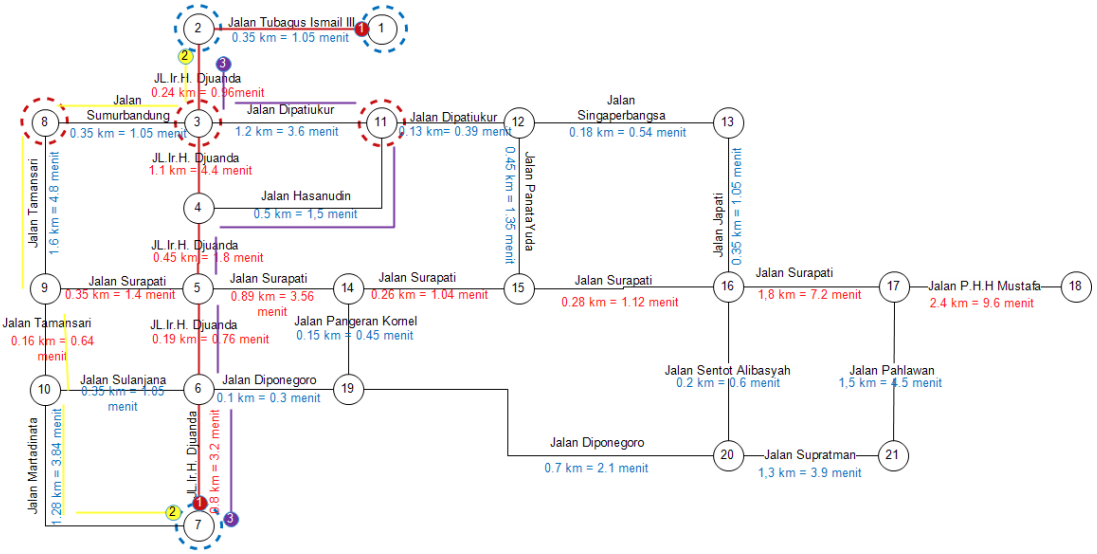
Gambar III. 35 Tahap Ke-3 dijkstra

Langkah 3 – memilih *node* terdekat dengan *node* sebelumnya C[1] = 0 < C[2] = 1.05. terpilih *node* 2 sebagai *node* saat ini selanjutnya.



Gambar III. 36 Tahap Ke- 4 dijkstra

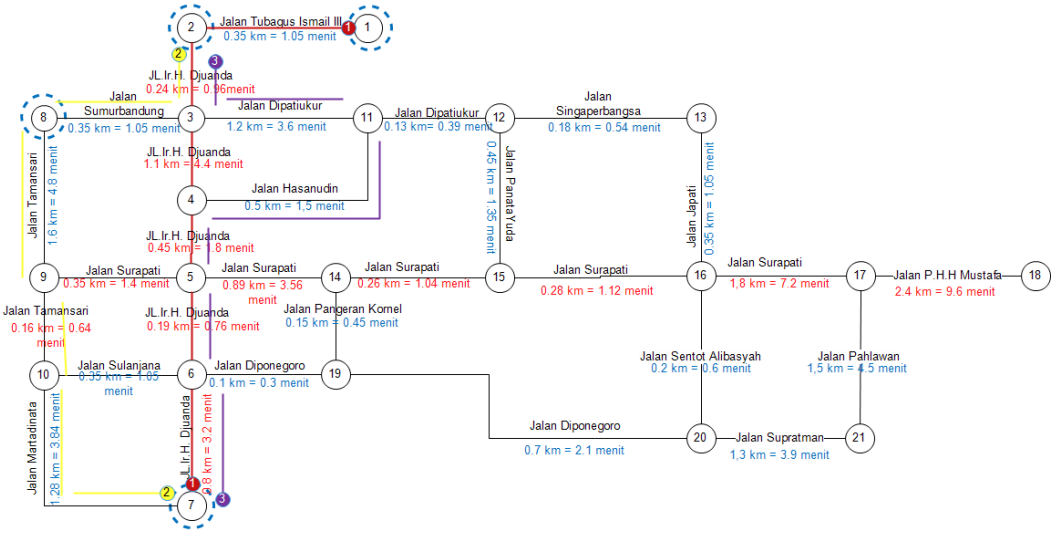
Langkah 4 - memperbarui C[K] untuk semua *node* terdekat dari *node* 2 yang tidak ada dalam {S}, Simpan *node* 2 sebagai *node* awal selanjutnya {S} = 1 , 2



Gambar III. 37 Tahap Ke- 5 dijkstra

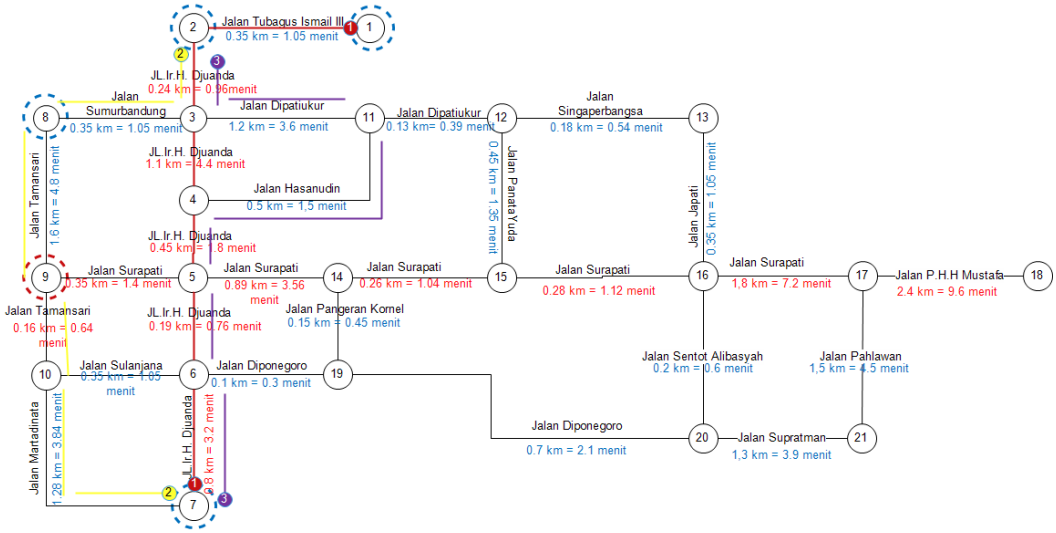
Langkah 5 – memilih *node* terdekat dengan *node* sebelumnya

C[2] = 1.05 C[3] = 2.01 , C[2] = 1.05 C[11] = 4.56, C[2] = 1.05 C[8] = 2.01, setelah dibandingkan maka yang nilai waktu terkecil terpilih *node* 8 sebagai *node* saat ini selanjutnya.



Gambar III. 38 Tahap Ke- 6 dijkstra

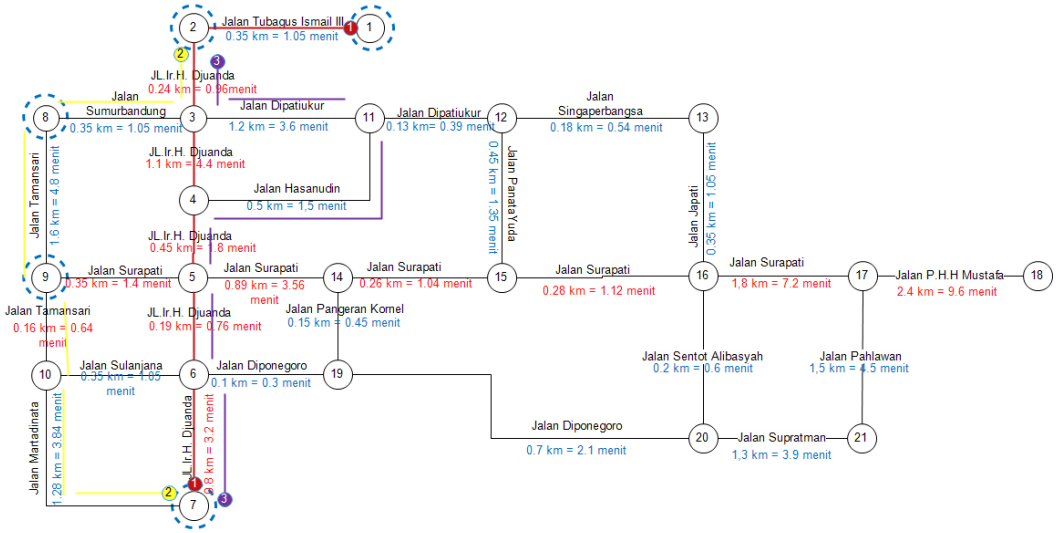
Langkah 6 - memperbarui C[K] untuk semua *node* terdekat dari *node* 3 yang tidak ada dalam {S}, Simpan *node* 8 sebagai *node* awal selanjutnya {S} = 1 , 2, 8



Gambar III. 39 Tahap Ke- 7 dijkstra

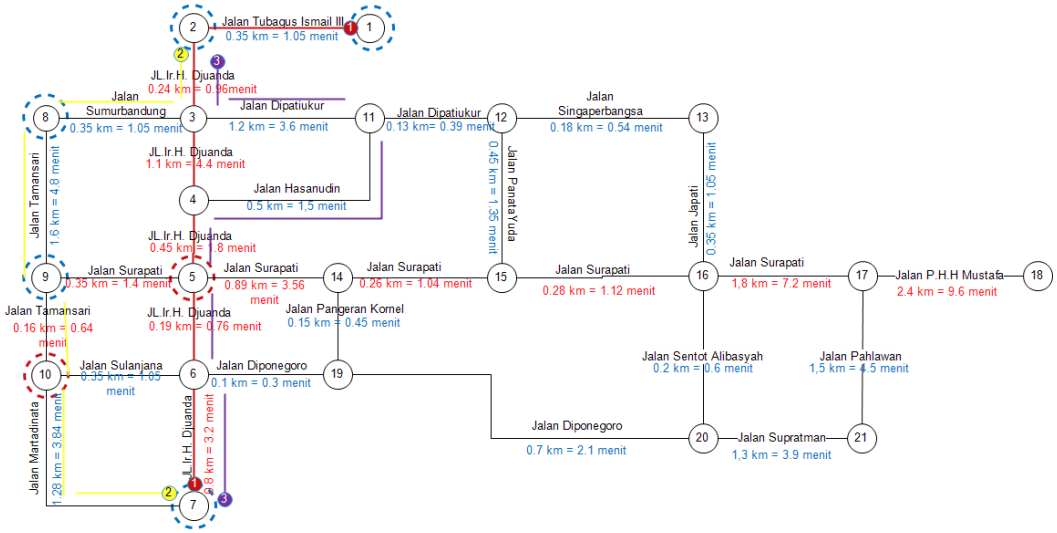
Langkah 7 – memilih *node* terdekat dengan *node* sebelumnya

C[8] = 2.01 C[9] = 4.8 , C[8] + C[9] = 6.81, setelah dibandingkan tidak terdapat *node* lain maka yang nilai waktu terkecil terpilih *node* 9 sebagai *node* saat ini selanjutnya.



Gambar III. 40 Tahap Ke- 8 dijkstra

Langkah 8 - memperbarui C[K] untuk semua *node* terdekat dari *node* 9 yang tidak ada dalam {S}, Simpan *node* 9 sebagai *node* awal selanjutnya {S} = 1 , 2, 8, 9

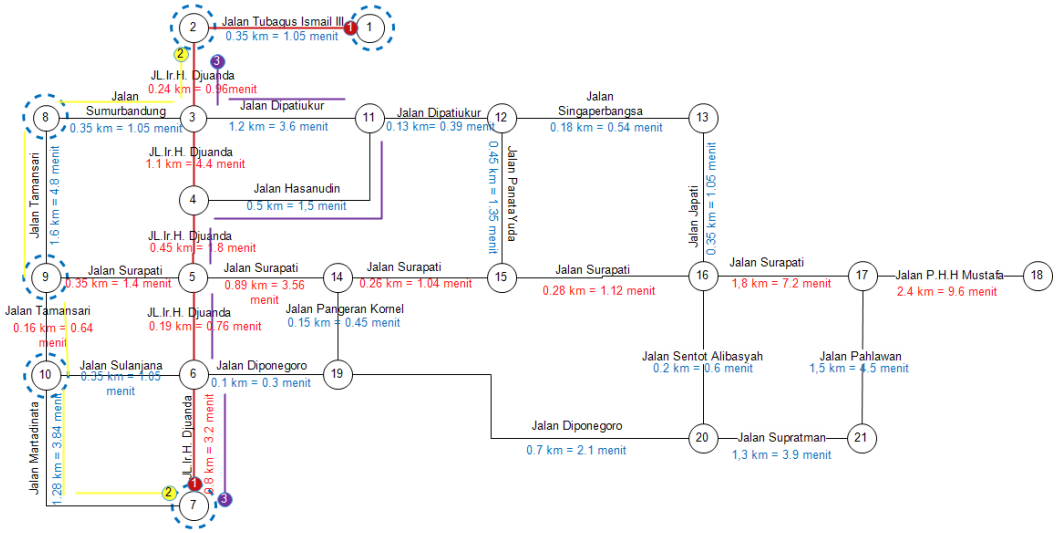


Gambar III. 41 Tahap Ke- 9 dijkstra

Langkah 9 – memilih *node* terdekat dengan *node* sebelumnya

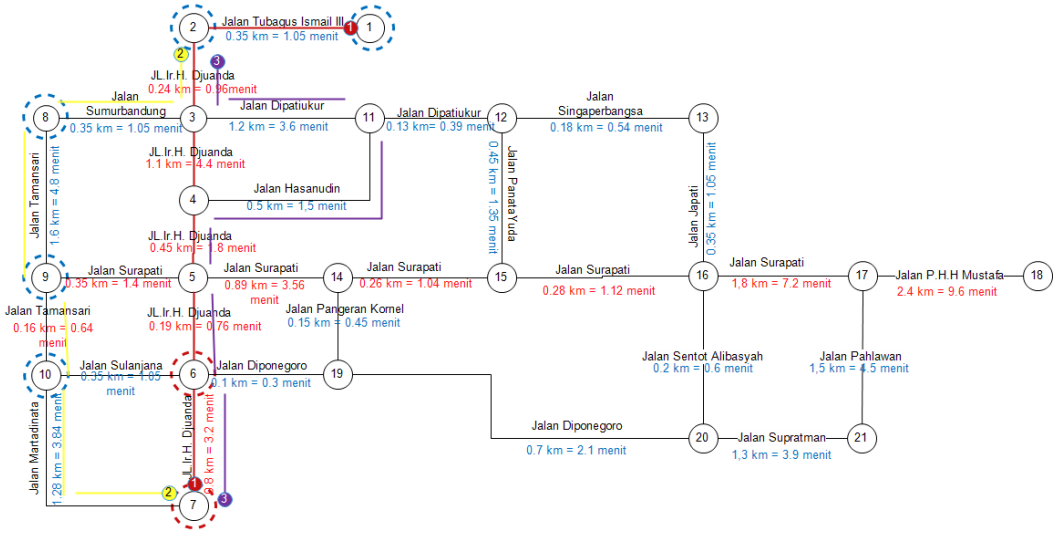
C[9] = 6.81, C[10] = 0.64 , C[8] + C[9] = 7.45,

C[9] = 6.81, C[5] = 1.4 , C[8] + C[9] = 8.21 setelah dibandingkan terdapat *node* lain maka nilai waktu terkecil terpilih *node* 10 sebagai *node* saat ini selanjutnya.



Gambar III. 42 Tahap Ke- 10 dijkstra

Langkah 10 - memperbarui C[K] untuk semua *node* terdekat dari *node* 10 yang tidak ada dalam {S}, Simpan *node* 10 sebagai *node* awal selanjutnya {S} = 1 , 2, 8, 9, 10



Gambar III. 43 Tahap Ke- 11 dijkstra

Langkah 11 – memilih *node* terdekat dengan *node* sebelumnya

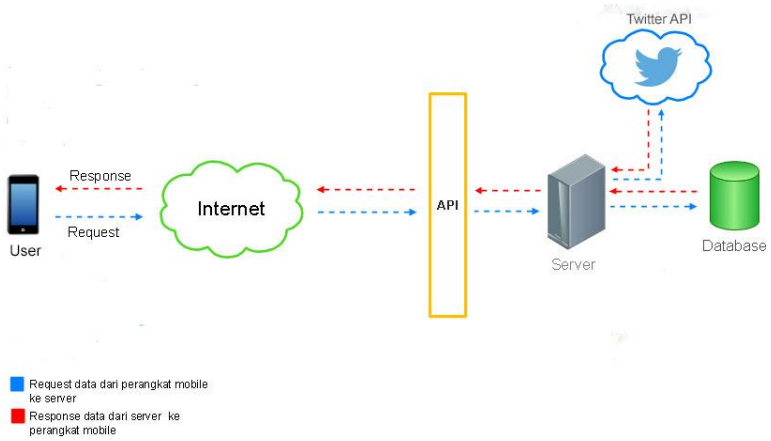
C[10] = 7.45, C[6] = 1.05 , C[8] + C[9] = 8.5,

C[10] = 7.45, C[7] = 3.84 , C[8] + C[9] = 11.29 setelah dibandingkan terdapat *node* 7 sebagai *node* tujuan, maka proses telah tercapai.

Jadi rute angkot terdekat adalah angkot 2 yang melewati Jalan Tubagus Ismail III – Jl. Ir. H. Djuanda – Jl. Sumur Bandung – Jl. Tamansari – Jl. Surapati – Jl. Tamansari – Jl. Martadinata dengan waktu tempuh 11 menit 29 detik.

III.6. Analisis Arsitektur Sistem

Platform mobile adalah salah satu subsistem yang dipilih untuk pembangunan dari perangkat lunak ini. Arsitektur perangkat lunak pada *platform mobile* mengambarkan bagaimana perangkat lunak saling berinteraksi seperti diilustrasikan pada Gambar III. 44 Arsitektur Perangkat Lunak Pada *Platform Mobile*. Gambar tersebut menggambarkan keseluruhan arsitektur sistem pada *platform mobile*.



Gambar III. 44 Arsitektur Perangkat Lunak Pada *Platform Mobile*

Berikut adalah deskripsi dari Gambar III. 44 Arsitektur Perangkat Lunak Pada *Platform Mobile*:

1. Perangkat *mobile* pengguna melakukan *request* data ke *server* melalui API
2. Perangkat *mobile* secara langsung akan memeberikan *request* positioning *mobile*.
3. *Server* menerima *request* data dari *server* dan menentukan jenis *request* yang diminta.
4. *Server* melakukan proses sesuai *request* ke *twitter* API
5. *Server* memberikan pesan sesuai *request* yang diberikan.

III.7. Spesifikasi Kebutuhan Perangkat Lunak

Spesifikasi kebutuhan perangkat lunak dibagi menjadi dua kebutuhan yaitu kebutuhan fungsional dan non fungsional. Spesifikasi kebutuhan fungsional dapat dilihat pada Tabel III. 14,dan spesifikasi kebutuhan non fungsional dapat dilihat pada Tabel III. 15.

Tabel III. 14 Spesifikasi Kebutuhan Fungsional

Nomor	Spesifikasi Kebutuhan Fungsional
SKPL-F-001	Aplikasi ini dapat membantu pengguna untuk mencari rute terdekat menuju lokasi tujuan menggunakan angkot.
SKPL-F-002	Aplikasi ini dapat membantu pengguna untuk mencari rute terdekat menuju lokasi tujuan menggunakan mobil.
SKPL-F-003	Aplikasi ini dapat membantu pengguna untuk mencari rute terdekat menuju lokasi tujuan menggunakan motor
SKPL-F-004	Aplikasi ini dapat membantu pengguna untuk mencari rute terdekat menuju lokasi tujuan menggunakan jalan kaki
SKPL-F-005	Aplikasi ini dapat pengguna untuk melihat info kemacetan
SKPL-F-006	Aplikasi ini dapat mencari peta lokasi tujuan sesui dengan *input*an pengguna

Tabel III. 15 Spesifikasi Kebutuhan Non Fungsional

Nomor	Spesifikasi Kebutuhan Non Fungsional
SKPL-NF-001	Sistem yang dibangun pada sistem *mobile*
SKPL-NF-002	Sistem yang dibangun minimal menggunakan sistem operasi android versi 4.2 Jelly Bean
SKPL-NF-003	Sistem yang dibangun menggunakan pemanfaatna perangkat GPS, dan A-GPS yang ada dalam *device*
SKPL-NF-004	Sistem dibangun dengan spesifikasi hardware yang memenuhi standar minimum kebutuhan
SKPL-NF-005	Desain *user interface* dan *user experience* pada perangkat lunak yang akan dibangun mengacu kepada *design guidelines* android dari google
SKPL-NF-006	Sistem tidak mengizinkan konten yang mengandung unsur sara, Pornografi dan norma-norma kesusilaan.
SKPL-NF-007	Perangkat lunak yang dibangun harus menghasilkan format file standar (.json) yang bisa digunakan oleh pihak luar yang berkepentingan.

III.8. Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non fungsional pada sistem ini meliputi analisis perangkat lunak, analisis perangkat keras dan perangkat pikir.

1. Product Requirement

Berikut adalah kebutuhan non fungsional berdasarkan *klasifikasi product requirement.*

a. SKPL-NF-001 Sistem yang dibangun sistem berbasis *mobile*.

b. SKPL-NF-002 Sistem yang dibangun minimal menggunakan sistem operasi android versi 4.2 Jelly Bean dan dibangun dengan dengan kebutuhan perangkat lunak minimum

   a) Sistem Operasi (*platform* bebas).

   b) *Web server* XAMPP versi 1.1.7 ke atas sebagai *web server*

c) Kode editor berupa *sublime text* 3.
d) *Integrated development environment* (IDE) berupa *Android Studio*.
e) *Android Development Tools* versi 22.1.6 sebagai *plugins Android Studio*.
f) Android *Software Development Kit* sebagai *framework* android

c. SKPL-NF-003 Sistem yang dibangun menggunakan perangkat GPS, dan A-GPS yang ada dalam *device*
d. SKPL-NF-004 Sistem dibangun dengan spesifikasi hardware yang memenuhi standar minimum kebutuhan seperti:
a) Komputer dengan VGA on board 512 MB
b) Layar dengan resolusi 1024x600
c) Komputer dengan ram 2 GB
d) Perangkat android dengan ram 512
e) *Device* android dengan resolusi 320x480

2. Organisational Requirement

Berikut adalah kebutuhan non fungsional berdasarkan klasifikasi *Organisational Requirement*.

a. SKPL-NF-005 Desain *user interface* pada perangkat lunak yang akan dibangun mengacu kepada *design guidelines* android dari google

3. External Requirement

Berikut adalah kebutuhan non fungsional berdasarkan klasifikasi *External Requirement.*

a. SKPL-NF-006 Sistem tidak mengizinkan konten yang mengandung unsur sara, pornografi dan norma-norma kesusilaan.
b. SKPL-NF-007 Perangkat lunak yang dibangun harus menghasilkan format *file* standar (.json) yang bisa digunakan oleh pihak luar yang berkepentingan.

4. Analisis Kebutuhan Non Fungsional Pengguna

Analisis kebutuhan non fungsional pengguna adalah batasan-batasan dari layanan-layanan dan fungsi-fungsi yang dibutuhkan dari sebuah sistem dari segi perangkat pikir. Dalam penelitian ini dikelompokkan menjadi 5 kategori yaitu *User Knowledge and Experience, User Job and Task Characteristic and User Physical Characteristic.*

a. *User Knowledge and Experience*

User Knowledge and Experience yang diharapkan dari pengguna perangkat lunak pada penelitian ini dapat dilihat pada Tabel III. 16.

Tabel III. 16 *User Knowledge and Experience*

Knowledge And Experiences	
Computer Literacy	Cukup sampai tinggi
Sistem Experiences	Rendah sampai tinggi
Application Experience	Rendah sampai tinggi
Task Experience	Rendah sampai tinggi
Other Sistem Use	*Frequent*
Education	SMA sampai Perguruan Tinggi
Reading Level	Sedang sampai Tinggi
Typing Skill	Sedang sampai Tinggi
Native Language Or Culture	*English*

b. *User Job and Task Characteristic*

User Job and Task Characteristic yang diharapkan dari pengguna perangkat lunak pada penelitian ini dapat dilihat pada Tabel III. 17.

Tabel III. 17 *Job and Task Characteristic*

Job and Task Characteristic	
Task Structure	Tinggi
Social Interactions	Perlu
Primary Training	*None*
Job Category	*None*
Frequency of Use	Rendah
Task or Need Importance	*Medium*

c. *User Physical Characteristic*

User Physical Characteristic yang diharapkan dari pengguna perangkat lunak pada penelitian ini dapat dilihat pada Tabel III. 18.

Tabel III. 18 *User Physical Characteristic*

User Physical Characteristic	
Age	*Young, middle aged, and elderly.*
Gender	*Male and Female*
Handedness	*Both*
Disabilities	*None*

III.9. Analisis Data

Analisis data yang digunakan dalam penelitian ini adalah *Object Relational Mapping* (ORM). *Object relational mapping* melakukan pemetaan terhadap tabel-tabel pada basis data relasional dengan suatu *class* entitas yang ada pada bahasa pemrograman berorientasi objek.

Pemetaan yang dilakukan ORM akan membutuhkan suatu jembatan berupa format data JSON yang dapat menghubungkan ORM dengan *database* fisik yang ada pada *server*. Struktur format data JSON yang digunakan pada penelitian ini terbagi menjadi tiga yaitu data objek, data *array* dan data *null*.

1. Data Objek

Format pada Tabel III. 19 digunakan ketika data yang diterima dari *server* merupakan data tunggal atau berupa satu objek

Tabel III. 19 Struktur JSON data objek

Struktur data objek

{
"key" : "value"
}

2. Data *Array*

Format pada Tabel III. 20 Struktur JSON *Array* digunakan ketika data yang diterima dari *server* merupakan data tunggal atau berupa satu objek.

Tabel III. 20 Struktur JSON data *array*

Struktur data *array*

{
"result" : "value",
"item" : [{ "key" : "value" },{ "key" : "value" }]
}

3. Data *Null*

Format Tabel III. 21 pada digunakan ketika data yang diterima dari *server* merupakan data tunggal atau berupa satu objek.

Tabel III. 21 Struktur Data *Null*

Struktur data *null*

{
"result" : "value",
"message" : "value",
"status_code":value
}

III.10.Analisis Kebutuhan Fungsional

1. *Use case diagram*

Use case diagram merupakan konstruksi untuk mendeskripsikan hubungan-hubungan yang terjadi antar aktor dengan aktivitas yang terdapat pada sistem. Sasaran pemodelan *use case* diantaranya adalah mendefinisikan kebutuhan fungsional dan operasional sistem yang akan dibangun.

Tabel III. 22 Definisi Aktor

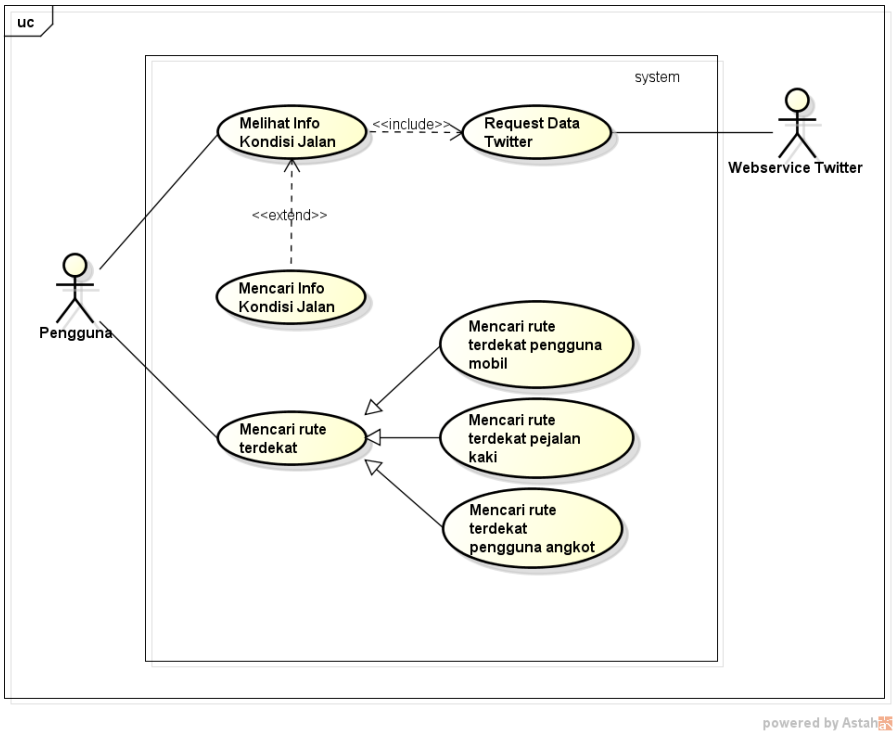
No	Aktor	Deskripsi
1	Pengguna	Pengguna dapat melakukan atau menggunakan fungsionalitas cari lokasi tujuan, melihat data *tweet* kemcetan, mencari rute terdekat menggunakan angkot, mencari rute terdekat menggunakan mobil, mencari rute terdekat mencari rute terdekat menggunkan jalan kaki.
2	*Webservice twitter*	*Webservice twitter* dapat melakukan pengiriman data *twitter*.

Definisi Definisi *Use case* dari pembangunan perangkat lunak ini dapat dilihat pada Tabel III. 23.

Tabel III. 23 Definisi *Use Case*

No	Usecase	Deskripsi
1	Melihat info kondisi jalan	Fungsionalitas ini digunakan oleh pengguna untuk melihat data kemacetan
2	Mencari info kondisi jalan	Fungsionalitas ini digunakan oleh pengguna untuk mencari info kondisi jalan
3	Resuest Data *Twitter*	Fungsional ini digunakan oleh *Webservice Twitter* untuk mengirim data *twitter*.
4	Mencari rute terdekat	Fungsionalitas ini digunakan oleh pengguna untuk mencari rute terdekat
2	Mencari rute terdekat pengguna mobil	Fungsionalitas ini digunakan oleh pengguna untuk mencari rute terdekat untuk pengguna mobil
3	Mencari rute terdekat pengguna pejalan kaki	Fungsionalitas ini digunakan oleh pengguna untuk mencari rute terdekat untuk pengguna pejalan kaki
4	Mencari rute terdekat pengguna angkot	Fungsionalitas ini digunakan oleh pengguna untuk mencari rute terdekat untuk pengguna angkot

Use case diagram implementasi teknologi LBS dalam membangun aplikasi pencarian rute terdekat menggunakan algoritma Dijkstra ini dapat dilihat pada Gambar III. 45.



Gambar III. 45 *Use case diagram* Aplikasi Pencarian Rute Terdekat

2. *Use case Scenario*

Use case Scenario merupakan deskripsi urutan langkah-langkah dalam proses bisnis baik yang dilakukan aktor terhadap sistem maupun yang dilakukan oleh sistem terhadap aktor. Berdasarkan *use case diagram* pada maka *use case scenario* untuk aplikasi pencarian rute terdekat ini terdiri dari *melihat info kondisi jalan, mencari info kondisi jalan, request data twitter, mencari rute terdekat pengguna mobil, mencari rute terdekat pejalan kaki, mencari rute terdekat pengguna angkot.*. Berikut ini penjelasan dari masing-masing *use case scenario use case scenario*.

1. *Use case Scenario* Melihat info kondisi jalan

Tabel III. 24 *Use case Scenario* Melihat info kondisi jalan

***Use case* name**	**Melihat info kondisi jalan**	
Related Requirements	-	
Goal In Context	Menampilkan nama jalan beserta info kondisi jalan.	
Preconditions	Membuka halaman info kondisi jalan	
Successful End Condition	Nama jalan dan info kondisi jalan dapat ditampilkan.	
Failed End Condition	Nama jalan dan info kondisi jalan gagal ditampilkan.	
Primary Actors	Pengguna	
Secondary Actors	**None**	
Trigger	Pengguna membuka halaman info kondisi jalan.	
Include Case	*Request* Data *Twitter*	
Main Flow	**Step**	**Action**
	1	Pengguna membuka halaman utama aplikasi
	2	Pengguna memilih menu "Info Kondisi Jalan"
	Include::*Request* Data *Twitter*	
	3	Sistem melakukan *crawling data*
	4	Data *crawling* tersimpan di *database*
Extension	**Step**	**Branching Action**
	2.1	Info kondisi jalan gagal di tampilkan.
	4.1	Data *crawling* gagal tersimpan

2. *Use case Scenario* Mencari info kondisi jalan

Tabel III. 25 *Use case Scenario* Mencari info kondisi jalan

***Use case* name**	**Mencari info kondisi jalan**	
Related Requirements	-	
Goal In Context	Menampilkan nama jalan beserta info kondisi jalan berdasarkan *keyword* yang di masukkan.	
Preconditions	Membuka halaman info kondisi jalan	
Successful End Condition	Nama jalan dan info kondisi jalan dapat ditampilkan.	
Failed End Condition	Nama jalan dan info kondisi jalan gagal ditampilkan.	
Primary Actors	Pengguna	
Secondary Actors	**None**	
Trigger	Pengguna menekan tombol pencarian pada halaman info kondisi jalan.	
Include Case	*Request* Data *Twitter*	
Main Flow	**Step**	**Action**
	1	Pengguna membuka halaman utama aplikasi
	2	Pengguna memilih menu "Info Kondisi Jalan"
	3	Pengguna memasukkan key*word* "nama jalan" dalam form pencarian
	4	Pengguna menekan *icon* cari.
Extension	**Step**	**Branching Action**
	4.1	Info kondisi jalan gagal di tampilkan.

3. *Use case Scenario Request* Data *Twitter*

Tabel III. 26 *Use case Scenario Request* data *twitter*

***Use case* name**	***Request* Data *Twitter***	
Related Requirements	-	
Goal In Context	Mengirim data *twitter* dari *server*.	
Successful End Condition	Data *twitter* berhasil dikirim	
Failed End Condition	Data *twitter* gagal dikirim	
Primary Actors	*Webservice Twitter*	
Secondary Actors	**None**	
Trigger	*Credential* dari *crawling* data sesuai dengan yang terdapat pada *webservice twitter*	
Include Case	*Request* Data *Twitter*	
Main Flow	**Step**	**Action**
	1	*Credential* dari *crawling* data sesuai dengan yang terdapat pada *webservice twitter*.
	2	*Credential crawling* data di verifikasi di*database*
	3	Data *twitter* yang di minta di kirim.
Extension	**Step**	**Branching Action**
	1.1	*Credential* dari *crawling* data tidak sesuai.

4. *Use case Scenario* Mencari rute terdekat pengguna mobil

Tabel III. 27 *Use case Scenario* Mencari rute terdekat pengguna mobil

***Use case* name**	**Mencari rute terdekat pengguna mobil**	
Related Requirements	-	
Goal In Context	Menampilkan jalur rute terdekat untuk pengguna mobil	
Preconditions	Pengguna mencari rute terdekat jika menggunakan mobil	
Successful End Condition	Jalur rute terdekat untuk pengguna mobil berhasil di tampilkan	
Failed End Condition	Jalur rute terdekat untuk pengguna mobil gagal di tampilkan	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol pencarian mobil	
Base Case	**Mencari rute terdekat**	
Main Flow	**Step**	**Action**
	1	Pengguna membuka halaman cari rute
	2	Pengguna memilih lokasi tujuan dan lokasi pengguna pada *textbox*
	3	Menampilkan data rute terdekat untuk pengguna mobil
Extension	**Step**	**Branching Action**
	3.1	Rute gagal ditampilkan

5. *Use case Scenario* Mencari rute terdekat pejalan kaki

Tabel III. 28 *Use case Scenario* Mencari rute terdekat pejalan kaki

***Use case* name**	**Mencari rute terdekat pejalan kaki**	
Related Requirements	-	
Goal In Context	Menampilkan jalur rute terdekat untuk pejalan kaki	
Preconditions	Pengguna mencari rute terdekat untuk pejalan kaki	
Successful End Condition	Jalur rute terdekat untuk pejalan kaki berhasil di tampilkan	
Failed End Condition	Jalur rute terdekat untuk pejalan kaki gagal di tampilkan	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol pejalan kaki	
Base Case	**Mencari rute terdekat**	
Main Flow	**Step**	**Action**
	1	Pengguna membuka halaman cari rute
	2	Pengguna memilih lokasi tujuan dan lokasi pengguna pada *textbox*
	3	Menampilkan data rute terdekat untuk pejalan kaki
Extension	**Step**	**Branching Action**
	3.1	Rute gagal ditampilkan

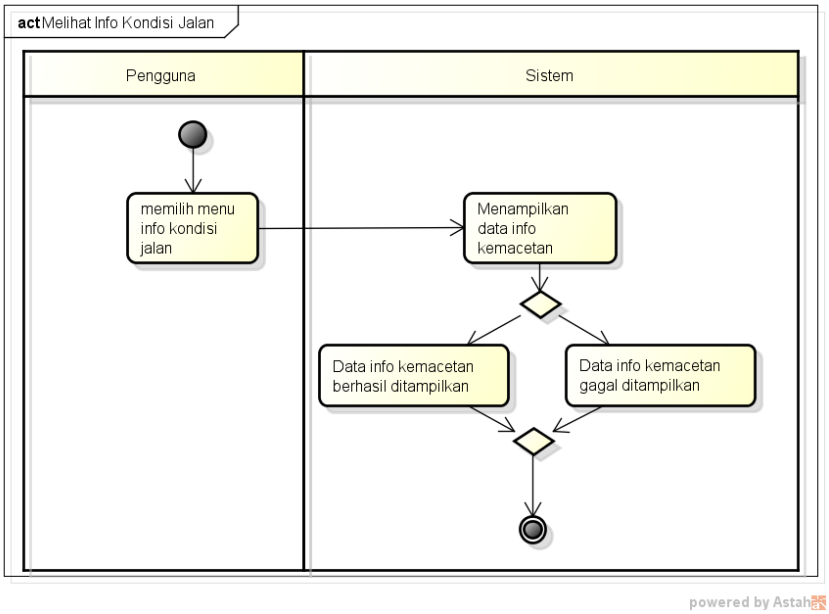
6. *Use case Scenario* Mencari rute terdekat pengguna angkot

Tabel III. 29 *Use case Scenario* Mencari rute terdekat pengguna angkot

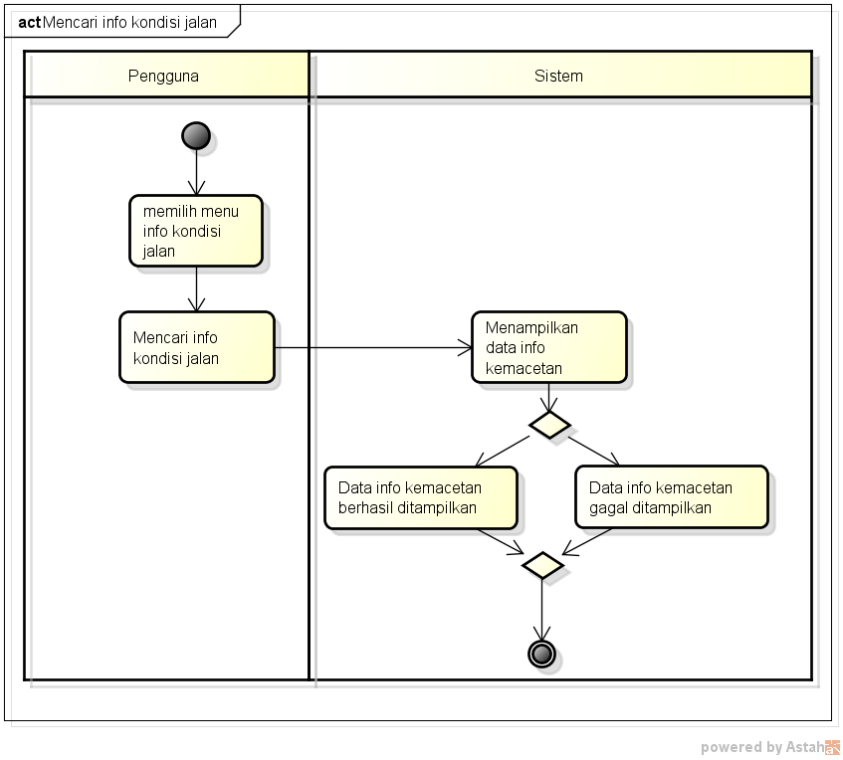
***Use case* name**	**Mencari rute terdekat pejalan kaki**	
Related Requirements	-	
Goal In Context	Menampilkan jalur rute terdekat untuk pengguna angkot	
Preconditions	Pengguna mencari rute terdekat untuk pengguna angkot	
Successful End Condition	Jalur rute terdekat untuk pengguna angkot berhasil di tampilkan	
Failed End Condition	Jalur rute terdekat untuk pengguna angkot gagal di tampilkan	
Primary Actor	Pengguna	
Trigger	Pengguna menekan tombol pengguna angkot	
Base Case	**Mencari rute terdekat**	
Main Flow	**Step**	**Action**
	1	Pengguna membuka halaman cari rute
	2	Pengguna memilih lokasi tujuan dan lokasi pengguna pada *textbox*
	3	Menampilkan data rute terdekat untuk pengguna angkot
Extension	**Step**	**Branching Action**
	3.1	Rute gagal ditampilkan

3. *Activity Diagram*

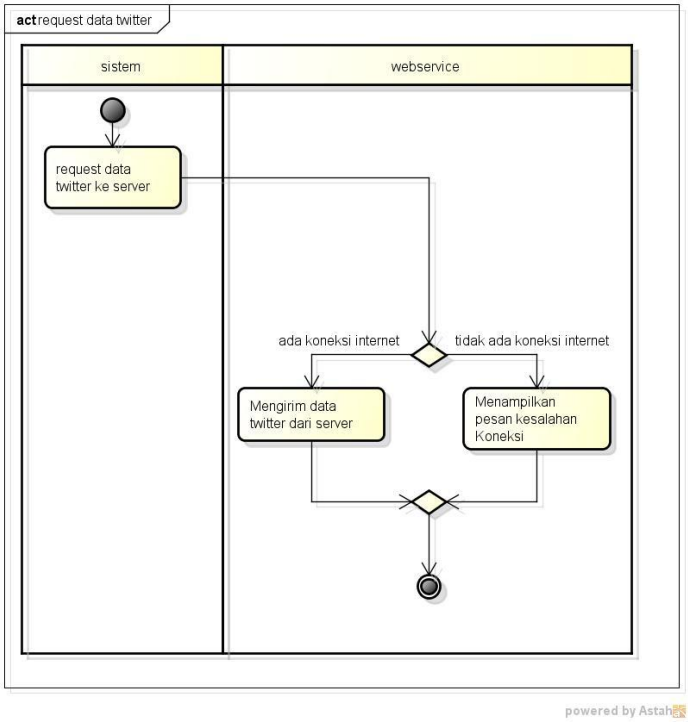
Activity Diagram dari pembangunan perangkat lunak pada penelitian ini adalah sebagai berikut:



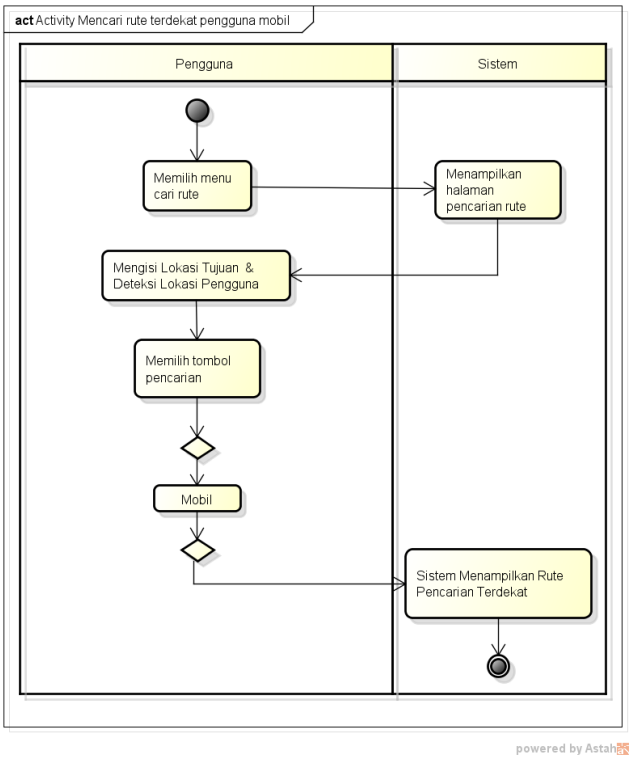
Gambar III. 46 *Activity* Melihat Info Kondisi Jalan



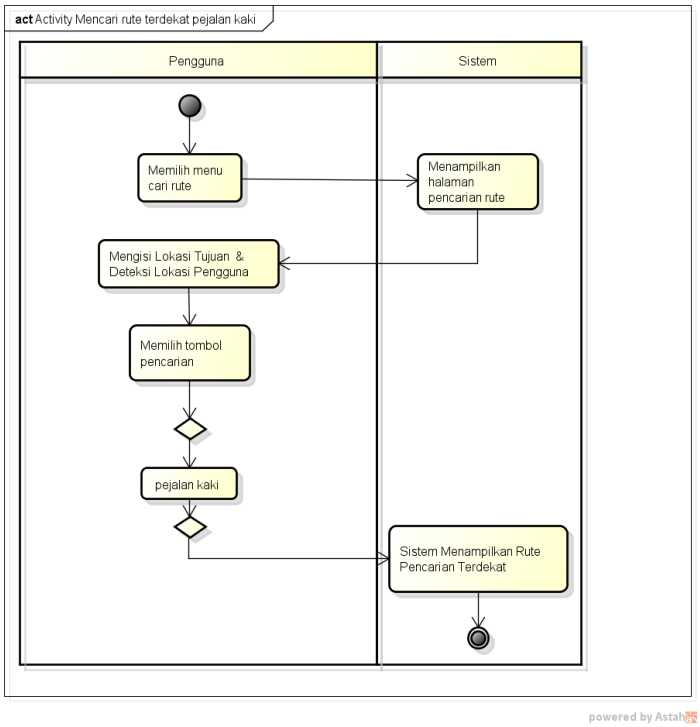
Gambar III. 47 *Acivity* Mencari Info Kondisi Jalan



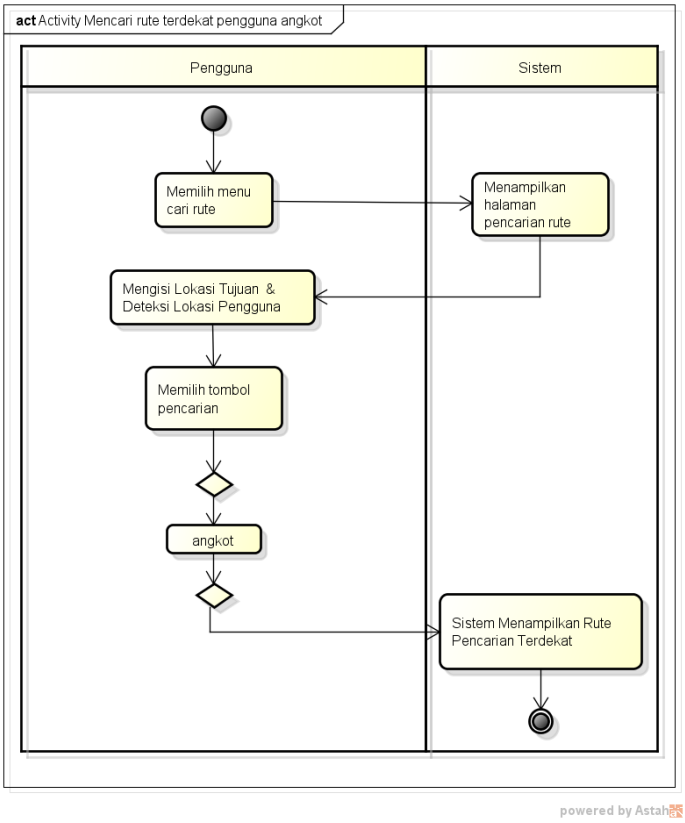
Gambar III. 48 *Activity Request* Data *Twitter*



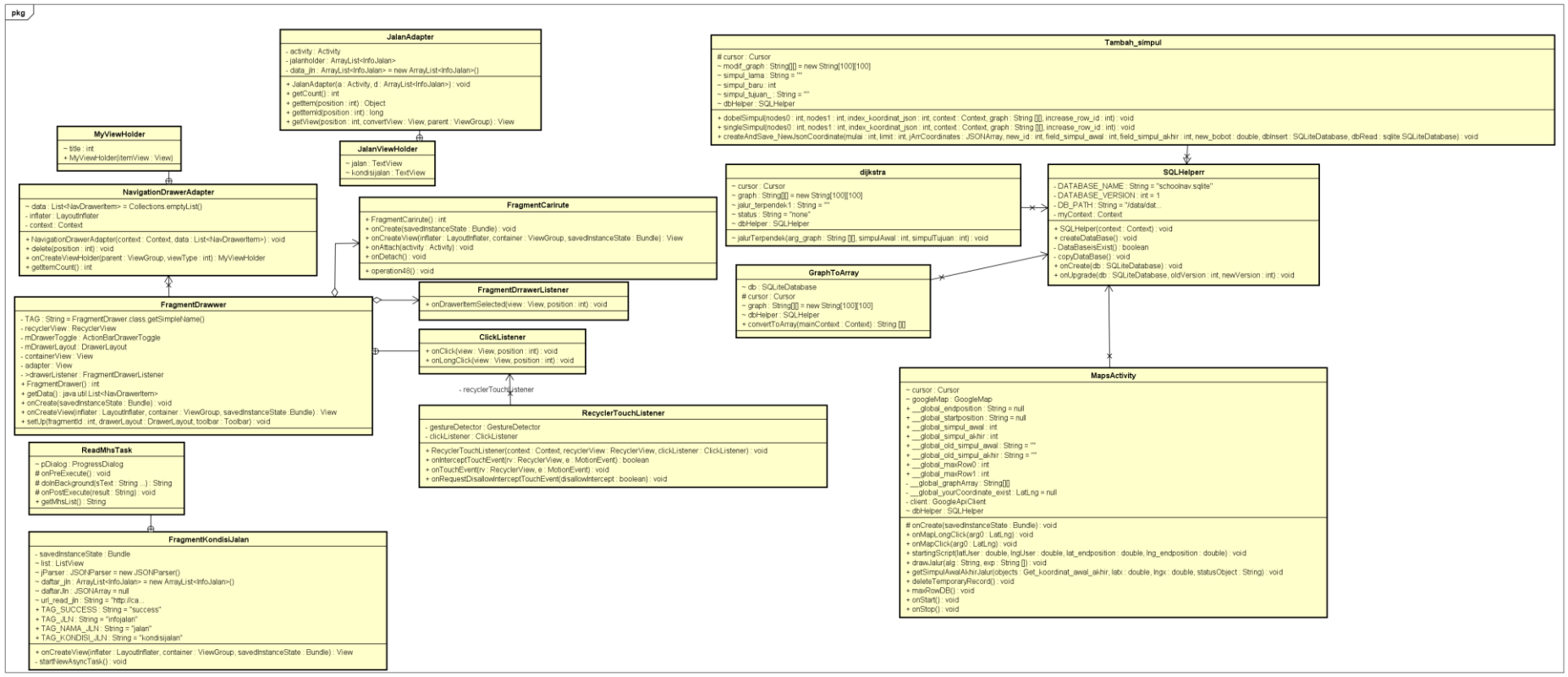
Gambar III. 49 *Activity* Mencari rute terdekat pengguna mobil



Gambar III. 50 *Activity* Mencari rute terdekat pejalan kaki



Gambar III. 51 *Activity* Mencari rute terdekat pengguna angkot

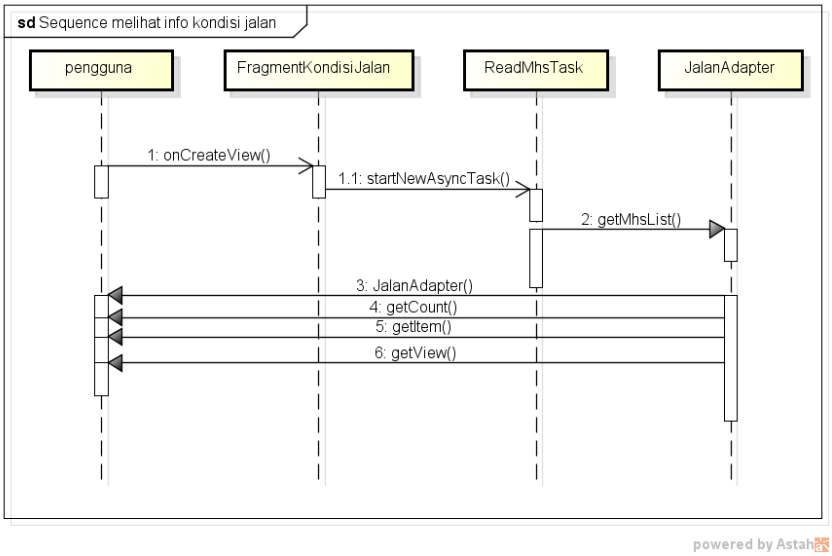


Gambar III. 52 *Class* Diagram

4. ***Sequence***

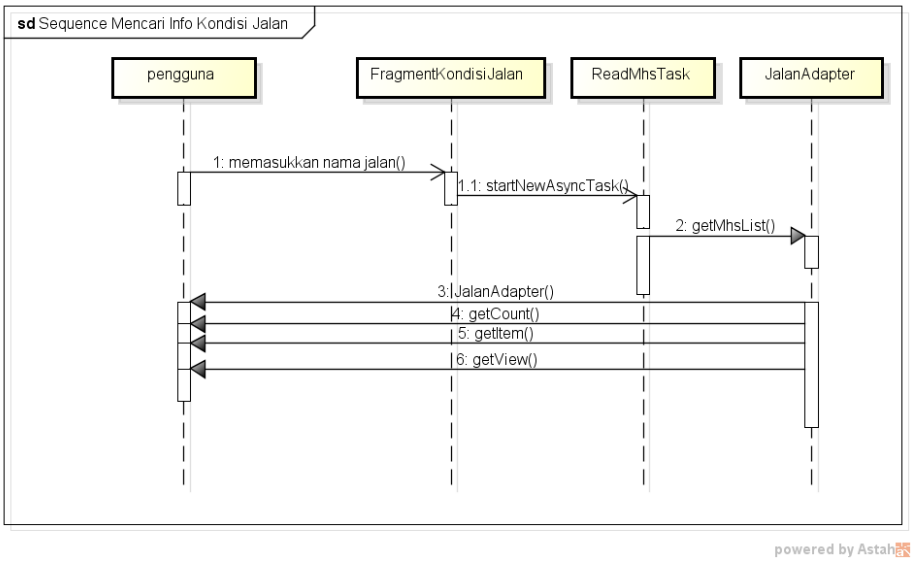
Sequence diagram digunakan untuk menggambarkan arus pekerjaan, pesan yang di sampaikan dan bagaimana elemen-elemen di dalamnya bekerja sama dari waktu ke waktu untuk mencapai suatu hasil.

1. *Sequence* Melihat Info Kondisi Jalan



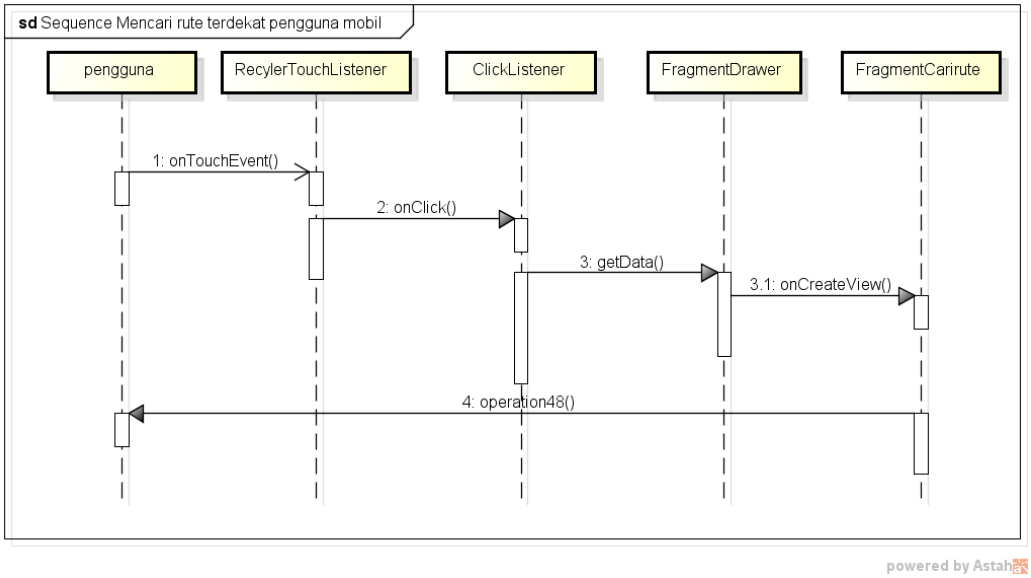
Gambar III. 53 *Sequence* Melihat Info Kondisi Jalan

2. *Sequence* Mencari Info Kondisi Jalan



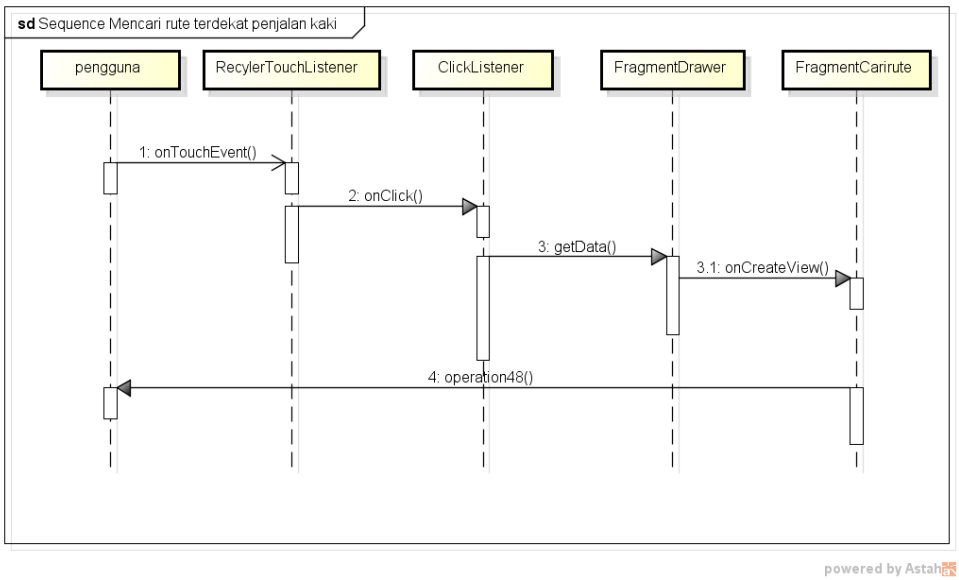
Gambar III. 54 *Sequence* Mencari Info Kondisi Jalan

3. *Sequence* Mencari Rute Terdekat Pengguna Mobil



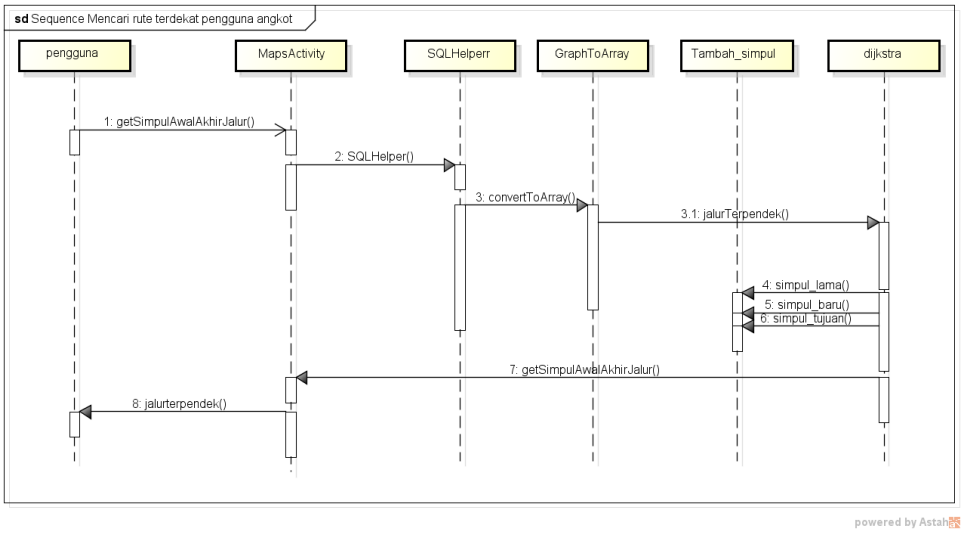
Gambar III. 55 *Sequence* Mencari Rute Terdekat Pengguna Mobil

4. *Sequence* Mencari Rute Terdekat Pejalan kaki



Gambar III. 56 *Sequence* Mencari Rute Terdekat Pejalan Kaki

5. *Sequence* Mencari Rute Terdekat Pengguna Angkot



Gambar III. 57 *Sequence* Mencari Rute Terdekat Pengguna Angkot

III.9 Perancangan Sistem

Pada perancangan Sistem ini, hal yang akan dibahas adalah Perancangan Data, Perancangan Arsitektural Menu, Perancangan Antarmuka, Perancangan Pesan dan Jaringan Semantik.

III.9.1.Perancangan Data

Pada penelitian ini *database* yang digunakan adalah MySQL maka akan ditampilkan struktur dari masing-masing collection yang akan digunakan:

a. Tabel *crawling*

Berikut adalah yang mendeskripsikan deskripsi tabel *crawling*.

Nama file: *crawling*

Tempat penyimpanan : *Database server*

Tabel III. 30 Tabel *Crawling*

Nama *Field*	Tipe Data	Panjang	Kunci	Keterangan
id_*tweet*	int	10	primary key	not *null*, auto increment
*tweet*_text	varchar	250		not *null*,
created	varchar	20		not *null*

b. Tabel klasifikasi

Berikut adalah yang mendeskripsikan deskripsi tabel klasifikasi.

Nama file: klasifikasi

Tempat penyimpanan : *Database server*

Tabel III. 31 Tabel Klasifikasi

Nama *Field*	**Tipe Data**	**Panjang**	**Kunci**	**Keterangan**
id_*tweet*	int	10	primary key, foreign key	not *null*, auto increment
*tweet*_text	varchar	250		not *null*,
kategori	varchar	17		not *null*

c. Tabel jalan

Berikut adalah yang mendeskripsikan deskripsi tabel jalan.

Nama file: jalan

Tempat penyimpanan : *Database server*

Tabel III. 32 Tabel Jalan

Nama *Field*	**Tipe Data**	**Panjang**	**Kunci**	**Keterangan**
id_jalan	int	10	primary key	not *null*, auto increment
nama_jalan	varchar	50		not *null*,
lat	varchar	30		not *null*,
lng	varchar	30		not *null*,
id_*tweet*	int	10	Foreign key	not *null*

d. Tabel detail *training*

Berikut adalah yang mendeskripsikan deskripsi tabel detail *training*.

Nama file: detail_*training*

Tempat penyimpanan : *Database server*

Tabel III. 33 Tabel Detail Training

Nama *Field*	Tipe Data	Panjang	Kunci	Keterangan
id_detail	int	10	primary key	not *null*, auto increment
kata	varchar	20		not *null*,
ProbMacetParah	varchar	50		not *null*
ProbMacet	varchar	50		not *null*
ProbLancar	varchar	50		not *null*

e. Tabel *stopword*

Berikut adalah yang mendeskripsikan deskripsi tabel *stopword*.

Nama file: *stopword*

Tempat penyimpanan : *Database server*

Tabel III. 34 Tabel *Stopword*

Nama *Field*	Tipe Data	Panjang	Kunci	Keterangan
id_*stopword*	int	11	primary key	not *null*, auto increment
stopword	varchar	30		not *null*,

f. Tabel kata

Berikut adalah yang mendeskripsikan deskripsi tabel kata.

Nama file: kata

Tempat penyimpanan : *Database server*

Tabel III. 35 Tabel Kata

Nama *Field*	Tipe Data	Panjang	Kunci	Keterangan
id_kata	int	10	primary key	not *null*, auto increment
kata	varchar	250		not *null*

g. Tabel data_latih

Berikut adalah yang mendeskripsikan deskripsi tabel data latih.

Nama file: data_latih

Tempat penyimpanan : *Database server*

Tabel III. 36 Tabel Data Latih

Nama *Field*	Tipe Data	Panjang	Kunci	Keterangan
id_*tweet*	int	11	primary key	not *null*, auto increment
*tweet*_text	varchar	180		not *null*,
kategori	varchar	20		not *null*

h. Tabel *preprocessing*

Berikut adalah yang mendeskripsikan deskripsi tabel *preprocessing*.

Nama file: *preprocessing*

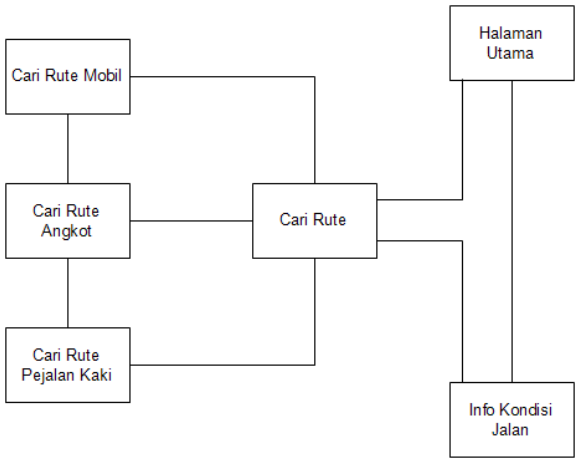
Tempat penyimpanan : *Database server*

Tabel III. 37 Tabel *Preprocessing*

Nama *Field*	Tipe Data	Panjang	Kunci	Keterangan
id_*tweet*	int	10	primary key	not *null*, auto increment
*tweet*_text	varchar	250		not *null*,

III.9.2.Perancangan Arsitektur Menu

Berikut adalah perancangan arsitektur menu dalam penelitian ini:



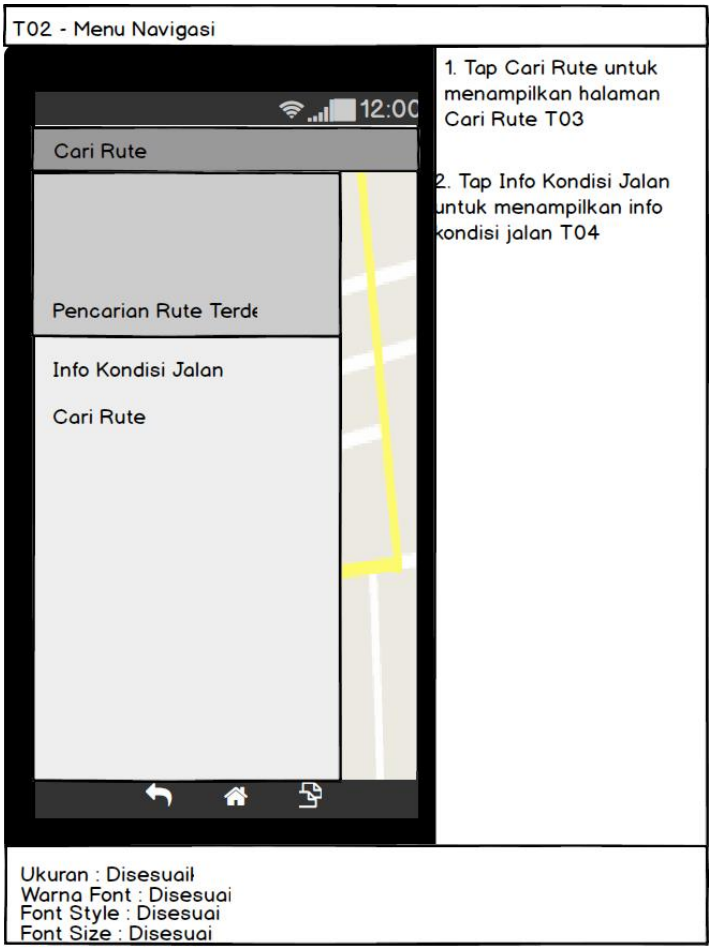
Gambar III. 58 Perancangan Arsitektur Menu

III.9.3.Perancangan Antar Muka

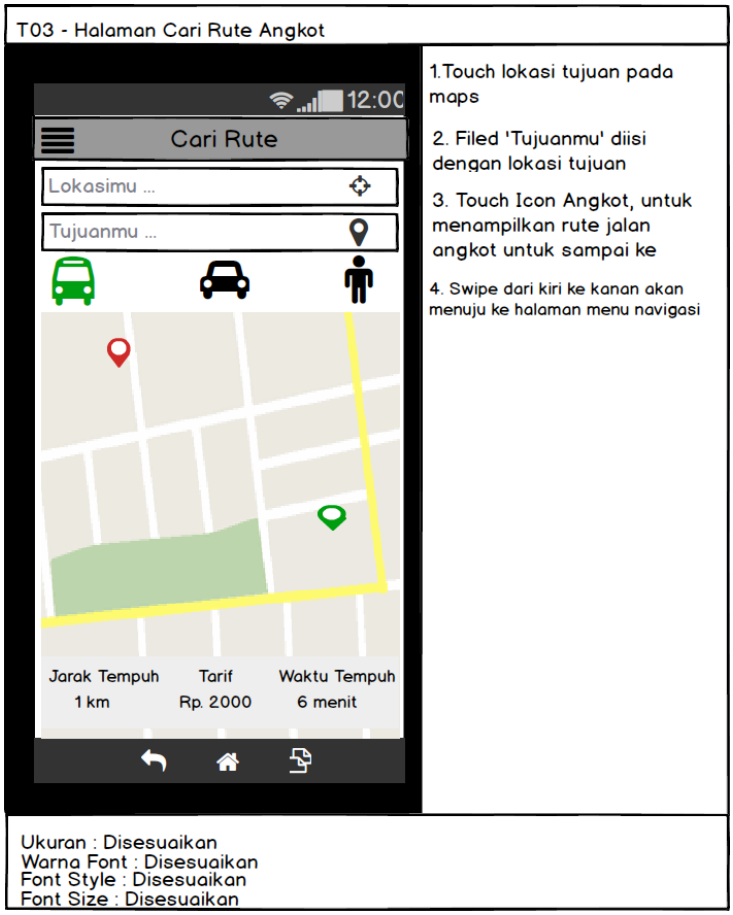
Berikut adalah perancangan antarmuka dalam penelitian ini:



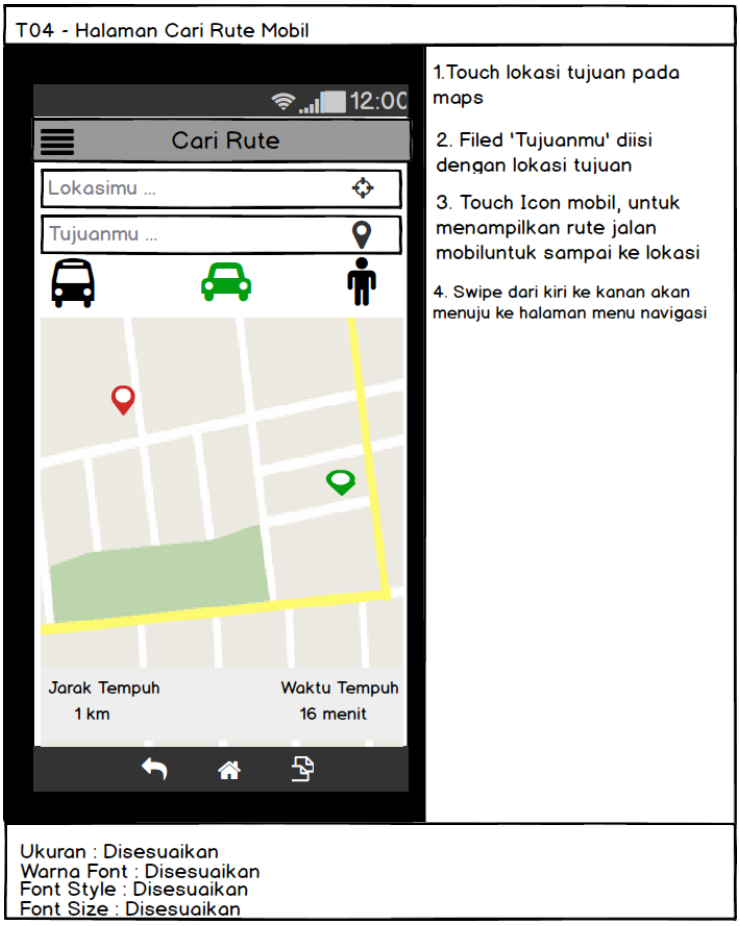
Gambar III. 59 Halaman Utama



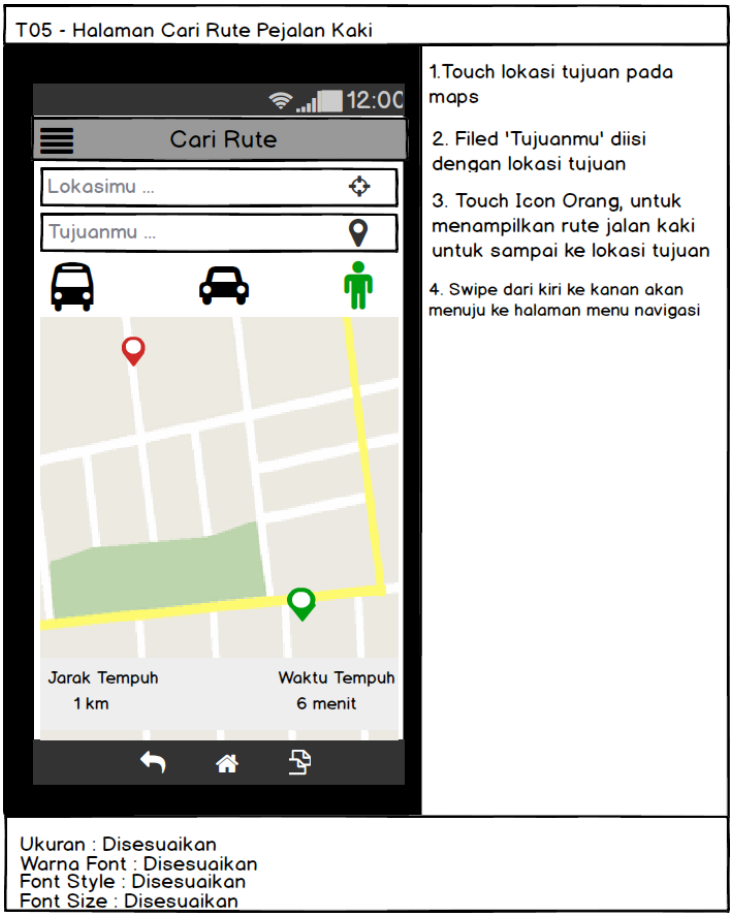
Gambar III. 60 Halaman Navigasi



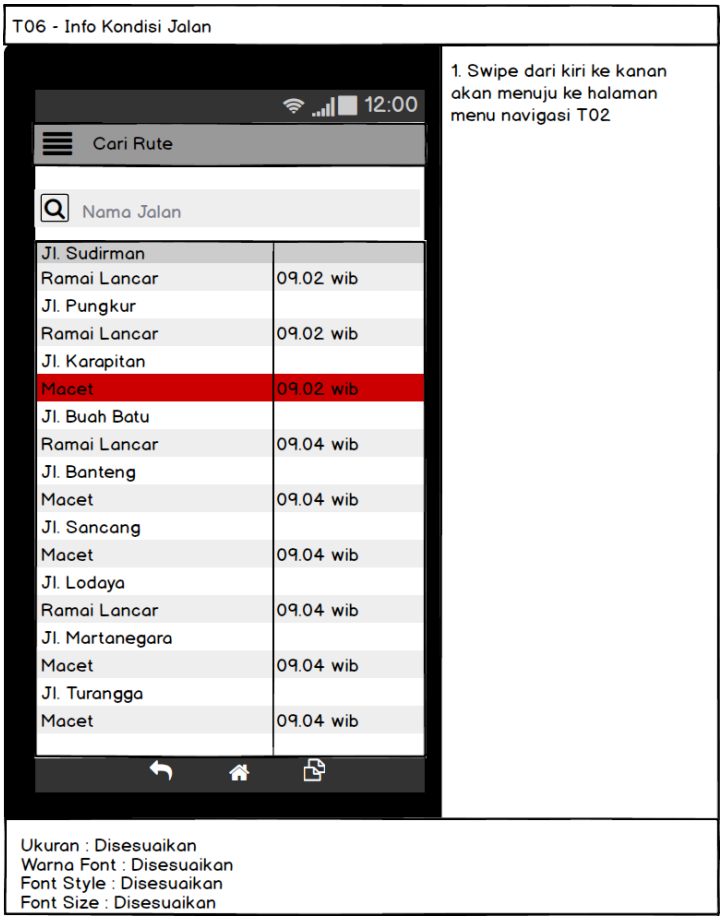
Gambar III. 61 Halaman Cari Rute Angkot



Gambar III. 62 Halaman Cari Rute Mobil



Gambar III. 63 Halaman Cari Rute Pejalan Kaki



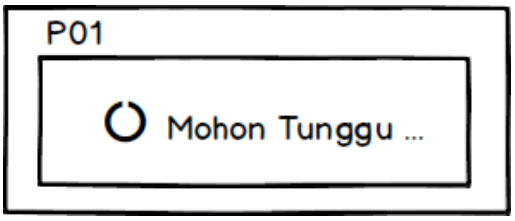
Gambar III. 64 Halaman Info Kondisi Jalan

III.9.4.Perancangan Pesan

Perancangan pesan pada perangkat lunak yang dibangun terbagi menjadi dua perancangan pesan *mobile* dan perancangan pesan web.

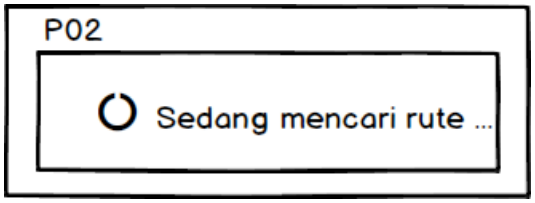
1. Perancangan Pesan *Mobile*

Berikut adalah perancangan pesan *mobile* pada perangkat lunak yang dibangun ketika menu info kondisi jalan di buka.



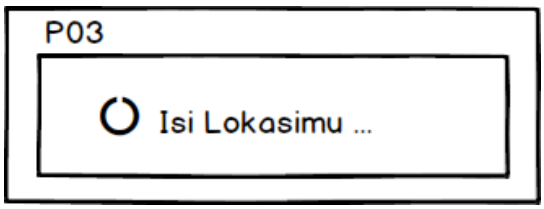
Gambar III. 65 Perancangan Pesan P01

Berikut adalah perancangan pesan *mobile* pada perangkat lunak yang dibangun ketika mencari rute.



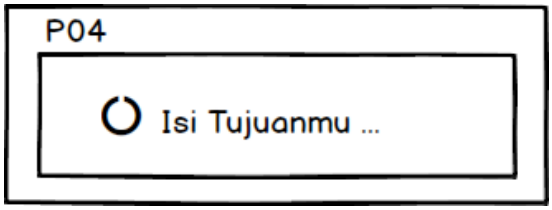
Gambar III. 66 Perancangan Pesan P02

Berikut adalah perancangan pesan *mobile* pada perangkat lunak yang dibangun ketika mencari rute saat lokasi belum diisi.



Gambar III. 67 Perancangan Pesan P03

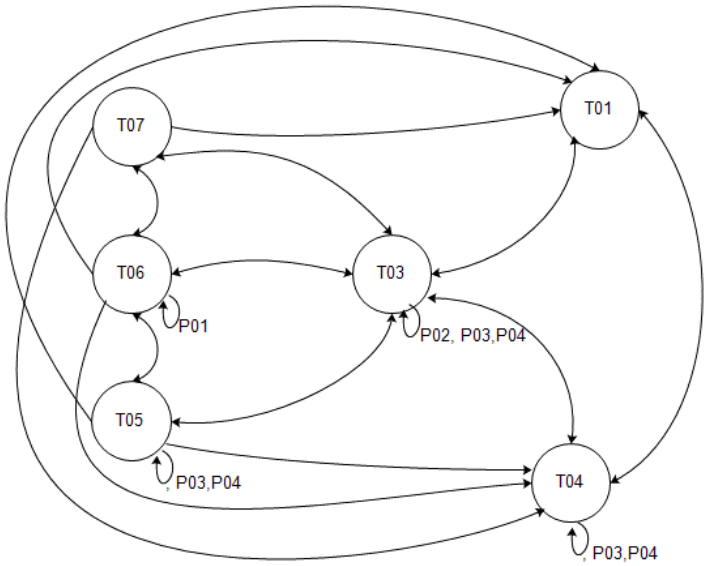
Berikut adalah perancangan pesan *mobile* pada perangkat lunak yang dibangun ketika mencari rute saat tujuan belum diisi.



Gambar III. 68 Perancangan Pesan P04

III.9.5.Jaringan Semantik

Jaringan Semantik merupakan gambaran pengetahuan grafis yang menunjukkan hubungan antar objek (lingkaran) dan garis yang menggambarkan formasi antarobjek tersebut. Adapun jaringan semantik untuk perangkat yang dibangun dapat dilihat pada Gambar III. 69.



Gambar III. 69 Jaringan Semantik