PERANCANGAN APLIKASI INFORMASI JADWAL DAN RUTE *BUS RAPID TRANSIT* TRANS SEMARANG BERBASIS SISTEM OPERASI ANDROID

Wibisono Indiarto [*)], and Maman Somantri

Departemen Teknik Elektro, Universitas Diponegoro Semarang
Jl. Prof. Sudharto, SH, Kampus UNDIP Tembalang, Semarang 50275, Indonesia

[*)] *E-mail: wibisono.indiarto@gmail.com*
***)Email : mmsomantri@live.undip.ac.id*

Abstrak

Transportasi umum merupakan salah satu kebutuhan masyarakat yang sangat diminati khususnya oleh masyarakat kota dalam kehidupan sehari-hari. Bus Rapid Transit Trans Semarang merupakan salah satu bentuk transportasi umum yang dapat dijumpai di kota Semarang. Adanya informasi tentang jadwal serta rute yang perlu ditempuh diperlukan demi kelancaran dan kenyamanan para pengguna bus. Kini pada era modern banyak sekali kebutuhan yang semakin dipermudah dengan menggunakan ponsel pintar untuk melengkapi kebutuhan-kebutuhan tersebut. Hal ini membuka peluang untuk mengetahui waktu keberangkatan bus dan rute mana yang perlu ditempuh para pengguna untuk dapat sampai ke tempat tujuan. Dengan layanan berbasis lokasi, pengguna akan dimudahkan dalam mencari informasi keberangkatan bus dan pencarian rute yang perlu ditempuh untuk sampai ke tempat tujuan. Hasil pengujian komunikasi data menunjukkan bahwa waktu yang diperlukan untuk mencari rute bus pada aplikasi rata-rata sebesar 5,2 detik.

Kata Kunci : Android, pencarian rute, BRT..

Abstract

Public transportation is a community needs which is favored especially by people in big cities on daily basis. Bus Rapid Transit Trans Semarang is one of many public transportations that can be found in the city of Semarang. The availability of information about bus schedules and routes that need to be taken is needed for bus users' comfort and ease. Nowadays on the modern era there are so many needs that getting more eased by using smartphones to fulfill those needs. This opens an opportunity to find out about bus' departure time and which route that needs to be taken to get to users' destination. Using location-based service, users are going to be eased in searching bus' departure time and route that needs to be taken to get to users' destination. Data communication test results show that average time needed for route finding in this application is 5.2 seconds.

Keyword: Android, route finding, BRT.

1. Pendahuluan

Transportasi umum merupakan salah satu kebutuhan yang sudah sangat diminati oleh masyarakat khususnya di kota-kota besar. Transportasi umum seperti bus dan kereta memiliki jumlah pengguna yang banyak karena selain dapat menampung banyak pnumpang juga memiliki tarif yang terjangkau oleh semua elemen masyarakat. Bus Rapid Transit Trans Semarang merupakan salah satu contoh transportasi umum yang sudah ada di kota Semarang.

Bus Rapid Transit Trans Semarang memiliki 6 (enam) koridor aktif yang tersebar di kota Semarang dan sekitarnya. Dengan adanya korido-koridor dengan tujuan yang berbeda-beda serta kurangnya informasi rute dapat membingungkan para pengguna yang belum terbiasa menggunakan Bus Rapid Transit Trans Semarang.

Salah cara untuk mengetahui jadwal keberangkatan dan rute bus adalah dengan melihat papan informasi yang tersedia pada halte bus. Namun, cara tersebut memiliki berbagai macam keterbatasan dan tidak selalu ada pada halte Bus Rapid Transit Trans Semarang.

Kini kebutuhan akan informasi dipermudah dengan aplikasi-aplikasi yang tersedia pada ponsel pintar dan Android merupakan sistem operasi terpopuler pada saat

ini. Berbagai macam aplikasi tersedia untuk memenuhi kebutuhan akan berbagai macam informasi dan dapat diunduh dengan mudah.

Salah satu fitur yang banyak digunakan pada aplikasi-aplikasi tersebut adalah layanan berbasis lokasi (Location-based Service) dimana layanan berbasis lokasi ini menggunakan teknologi Global Positioning Service dan Cell-based Location dari Google.

Salah satu algoritma pencarian rute yang banyak digunakan adalah algoritma Dijkstra. Algoritma Dijkstra akan mencari rute terpendek dari satu titik ke titik yang lain dalam suatu graf satu per satu hingga kondisi tersebut terpenuhi[1]. Algoritma Dijkstra digunakan untuk mencari rute yang perlu ditempuh pengguna dengan bobot terkecil, dimana bobot yang diperhitungkan adalah jumlah *hop* antar dua halte.

Penelitian mengenai pencarian rute menggunakan algoritma Dijkstra pada dasarnya telah dikembangkan pada sistem operasi Android[2]. Pada penelitian tersebut, perancangan aplikasi hanya dikhususkan untukpengguna masyarakat umum saja. Desain tampilan dan fitur yang dimiliki aplikasi tersebut masih sangat sederhana, sebatas mencari rute antar sepuluh tempat wisata. Untuk media penyimpanan, aplikasi tersebut masih menggunakan SQLite yang terdapat pada android, sehingga seluruh data akan tersimpan ke dalam memori *smartphone.* Penggunaan SQLite masih memiliki kekurangan yaitu, penyimpanan data dalam jumlah besar akan membebani kinerja *smartphone* dikarenakan kapasitas penyimpanan yang terbatas.

Penelitian lain tentang penggunaan algoritma Dijkstra pada pencarian rute bus juga telah dikembangkan[3]. Pada penelitian tersebut, data telah disimpan pada *server* sehingga memudahkan untuk pengembangan sistem. Namun, sistem operasi pada sisi *client* belum menggunakan sistem operasi Android.

Penelitian mengenai aplikasi pencarian rute BRT Tran Semarang pada sistem operasi Android juga sudah pernah dikembangkan[4]. Pada penelitian tersebut, aplikasi sudah menggunakan algoritma Dijkstra. Namun, sistem hanya terbatas pada pengguna jasa layanan BRT Trans Semarang.

Penelitian mengenai penerapan algoritma Dijkstra untuk pencarian rute menggunakan Object Oriented Programming juga telah dikembangkan[5]. Pada penelitian tersebut, pencarian rute menggunakan algoritma Dijkstra menggunakan aplikasi berbasis web. Terdapat juga fitur admin untuk mengelola shelter, jalur dan koridor.

Berdasarkan permasalahan tersebut, maka muncul ide untuk memanfaatkan algoritma Dijkstra dan *Location-based* service untuk menampilkan informasi keberangkatan bus dan rute yang perlu ditempuh oleh pengguna. Dengan memanfaatkan teknologi ini, diharapkan dapat menutupi kelemahan yang ada pada sistem informasi jadwal keberangkatan dan rute bus saat ini.

2. Metode

2.1. Metodologi Penelitian

Tahapan pelaksanaan penelitian untuk menyelesaikan Penelitian ini adalah sebagai berikut:

1. Studi Literatur
   Studi literatur dilakukan dengan mempelajari permasalahan yang akan dikemukakan pada Penelitian melalui beberapa buku literatur, dan menganalisa data menggunakan tulisan yang berhubungan dengan sistem yang dirancang, baik dari perpustakaan, artikel, maupun internet.

2. Perancangan Sistem
   Perancangan sistem dilakukan menggunakan metode *Waterfall* dengan tahapan metode berikut:
   a. Melakukan identifikasi terhadap kebutuhan sistem berdasarkan studi literatur.
   b. Menganalisis kebutuhan sistem kemudian menggambarkannya dalam bentuk definisi kebutuhan spesifik.
   c. Menggunakan definisi kebutuhan untuk mendesain sistem dengan memetakan kebutuhan sistem kedalam bentuk desain sistem termasuk perancangan gambaran aplikasi dan relasi di dalam sistem.
   d. Mengkodekan sistem menjadi aplikasi yang dapat dijalankan oleh *smartphone* berdasarkan desain sistem.
   e. Menguji aplikasi yang telah dibuat untuk memastikan aplikasi telah sesuai dengan definisi kebutuhan dan berfungsi dengan benar. Pengkodean ulang dilkukan pada saat aplikasi tidak berjalan seperti seharusnya. Pengujian Sistem
3. Pengujian yang dilakukan pada sistem meliputi:
   a. Pengujian *white box* pada aplikasi .BRT dan .PETUGAS berbasis android, dilakukan dengan menggunakan metode *basis path* untuk menguji alur logika pada fitur inti aplikasi .BRT dan .PETUGAS berbasis android.
   b. Pengujian komunikasi data, dilakukan dengan mengimplementasikan alamat tujuan dan model pengiriman pada aplikasi hurl.it.
   c. Pengujian kompabilitas perangkat keras, dilakukan dengan menguji secara langsung kepada beberapa tipe ponsel pintar.

2.2. Analisis Kebutuhan

2.2.1. Deskripsi Sistem

Dalam penelitian ini, penulis membuat dua buah aplikasi informasi jadwal dan rute Bus Rapid Transit pada ponsel

pintar berbasis Android. Konsep yang dibahas adalah bagaimana aplikasi client yang tertanam pada ponsel android dapat melakukan komunikasi dengan database server dengan menggunakan web service. Sistem ini terdiri dari dua aplikasi berbasis sistem operasi android dan sebuah server. Pembuatan dua aplikasi disini dikarenakan adanya dua jenis user, yaitu pengguna dan petugas yang memiliki fitur yang berbeeda. Hal ini dapat dimanfaatkan untuk mengurangi penyalahgunaan aplikasi untuk penyebaran informasi yang tidak benar.

Pengguna dapat mengakses jadwal bus yang sedang beroperasi, melihat halte bus yang berada di sekitar pengguna dan dapat mencari rute yang perlu ditempuh. Untuk melihat jadwal bus, pengguna dapat melihat jadwal dengan cara memilih halte pada daftar halte atau memilih marker pada peta. Untuk melihat halte yang berada di sektiar pengguna, pengguna akan dihadapkan pada tampilan peta yang berisi marker halte yang ada pada trayek BRT Trans Semarang. Pada pencarian rute, pengguna akan dihadapkan pada tampilan pemilihan halte asal dan halte tujuan. Setelh memilih halte asal dan halte utjuan, sistem akan mencari rute terdekat untuk pengguna dan kemudian akan ditampilkan dalam bentuk daftar halte.

Petugas dapat melakukan login, check-in pada halte dan logout. Ketika login, petugas perlu memasukkan id, password dan nomor bus yang akan dioperasikan. Setelah login berhasil, pengguna akan diarahkan menuju halamn check-in halte. Pada halaman tesebut, petugas dapat melakukan check-in pada halte yang sedang dilewati untuk memperbarui jadwal bus yang sedang beroperasi. Ketika petugas menekan tombol logout, petugas akan diarahkan kembali ke halaman login dan data jadwal bus yang dioperasikan oleh petugas akan dihapus.

2.2.2. Kebutuhan Fungsional

Kebutuhan fungsional merupakan gambaran mengenai fungsi-fungsi yang dapat dilakukan oleh sistem ini. Aplikasi yang dirancang adalah sebuah aplikasi informasi jadwal dan rute BRT pada perangkat bergerak berbasis android. Kebutuhan fungsional sistem meliputi:

1. Pengguna dapat melihat posisi halte-halte di sekitarnya dalam peta.
2. Pengguna dapat mencari informasi tentang halte dan jadwal bus yang melewati halte tersebut.
3. Pengguna dapat mencari rute yang perlu ditempuh untuk menuju ke suatu tempat.
4. Pengguna dapat mengetahui lokasinya secara real-time
5. Petugas dapat melakukan check-in pada halte untuk memperbarui posisi bus.
6. Petugas dapat mengaktifkan dan menon-aktifkan status bus.

2.2.3. Kebutuhan Non Fungsional

Kebutuhan non-fungsional adalah kebutuhan sistem meliputi kinerja, kelengkapan operasi pada fungsi-fungsi yang ada, serta kesesuaian dengan lingkungan penggunanya. Kebutuhan non-fungsional ini melingkupi beberapa kebutuhan yang mendukung kebutuhan fungsional, rumusan kebutuhan non-fungsional meliputi:

a. Kebutuhan Keamanan
   1. Aplikasi untuk Petugas mempunyai halaman untuk login agar hanya dapat diakses oleh Petugas yang berwenang.
   2. Menggunakan mekanisme enkripsi password MD5 sebagai sistem keamanan (verifikasi petugas).
   3. Setiap petugas hanya dapat login pada satu perangkat.

b. Kebutuhan Operasional
   1. Aplikasi untuk Pengguna dapat diunduh secara bebas melalui internet.
   2. Aplikasi untuk petugas dapat diperoleh melalui pihak yang bertanggung jawab terhadap operasional bus.
   3. Sistem ini dibuat menggunakan bahasa pemrograman Java.
   4. Sistem mennggunakan pertukaran data dengan format JSON.
   5. Pengguna hanya dapat melihat dan mencari informasi jadwal dan rute bus.

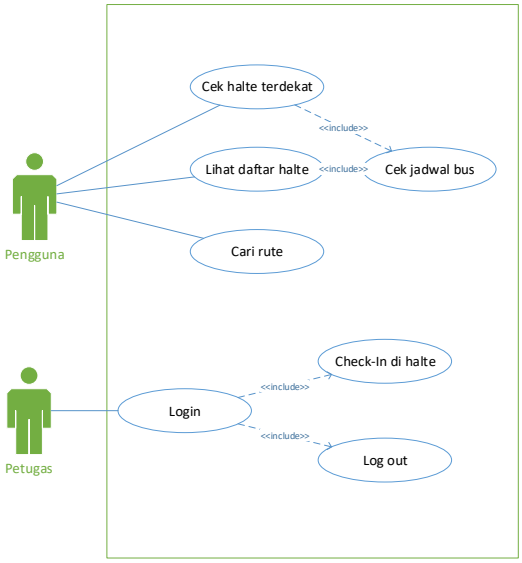
2.2.4. Kebutuhan Perangkat

Dalam pembuatan aplikasi ini, dibutuhkan suatu perangkat keras dan perangkat lunak. Kebutuhan perangkat keras meliputi *Notebook* Dell Inspiron N4050 dengan spesifikasi Intel(R) Core(TM) i3-2310M CPU @ 2.10 GHz, memori sebesar 4096MB, dan kapasitas *harddisk* sebesar 500 GB. Pada perangkat lunak dibutuhkan Android Studio 2.3 sebagai IDE dalam penulisan kode aplikasi Informasi Jadwal dan Rute BRT, dan Visual Paradigm v.13.0 dan Microsoft Visio untuk desain pemodelan objek.

2.3. Desain Aplikasi

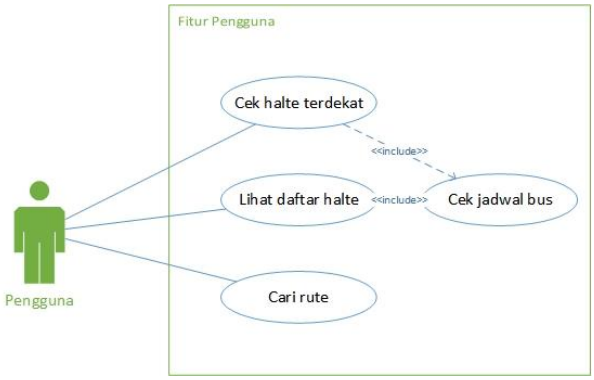
2.3.1. Diagram *Use Case*

Diagram *use case* menggambarkan fungsi-fungsi yang ada pada sistem. Diagram ini lebih berfokus pada fitur-fitur sistem dari sudut pandang pihak luar, yang dalam hal ini adalah pengguna dan petugas. Gambar 1 merupakan *use case* semu aktor dalam aplikasi .BRT dan .PETUGAS yaitu pengguna dan petugas.. Gambar 2 merupakan diagram *use case* pengguna aplikasi sistem presensi berbasis android sedangkan Gambar 3 merupakan *use case* petugas.



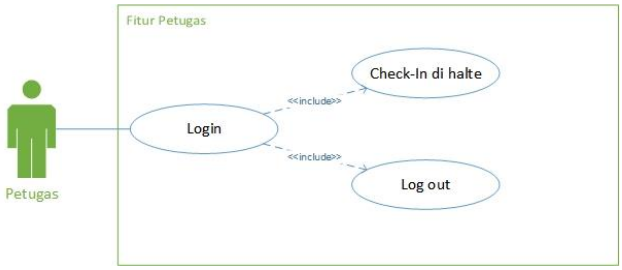
Gambar 1. Diagram *use case* aplikasi .BRT dan .PETUGAS

Pada Gambar 1 terlihat ada 2 jenis *user* yaitu pengguna dan petugas. *User* pengguna pada Gambar 1 adalah pengguna jasa layanan BRT Trans Semarang sedangkan *user* petugas adalah petugas operasional jasa layanan BRT Trans Semarang.



Gambar 2. Diagram *use case* pengguna aplikasi .BRT

Pada Gambar 2 terlihat bahwa fitur aplikasi dilihat dari sudut pandang pengguna aplikasi terdiri dari cari rute, lihat daftar halte, cek halte terdekat dan cek jadwal bus.



Gambar 3. Diagram *use case* petugas aplikasi .PETUGAS

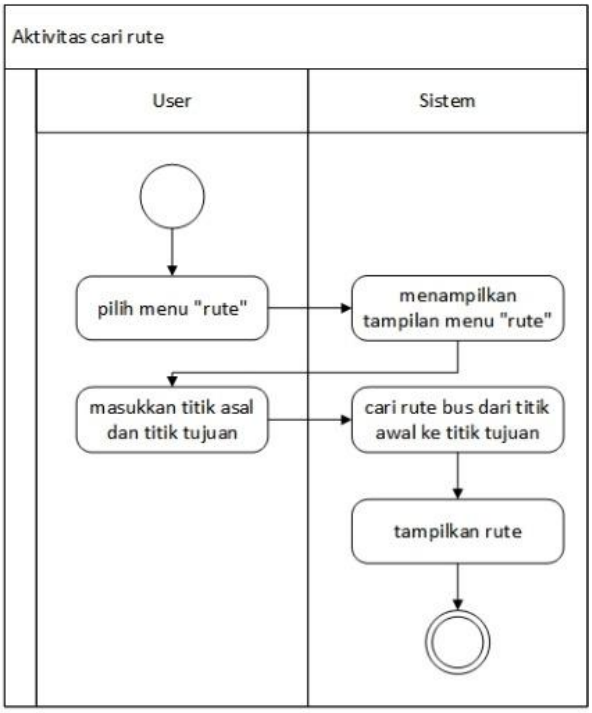
Pada Gambar 3 terlihat bahwa fitur aplikasi dilihat dari sudut pandang petugas terdiri dari *login*, *check-in* halte dan *log out*.

Fitur-fitur pengguna aplikasi dan petugas merupakan fitur-fitur berbeda yang dikhususkan untuk masing-masing aktor. Hal ini bertujuan untuk menghindari adanya masalah keamanan pada sistem yang dapat mengakibatkan penyebaran informasi yang tidak benar.

2.3.2. Diagram Aktivitas

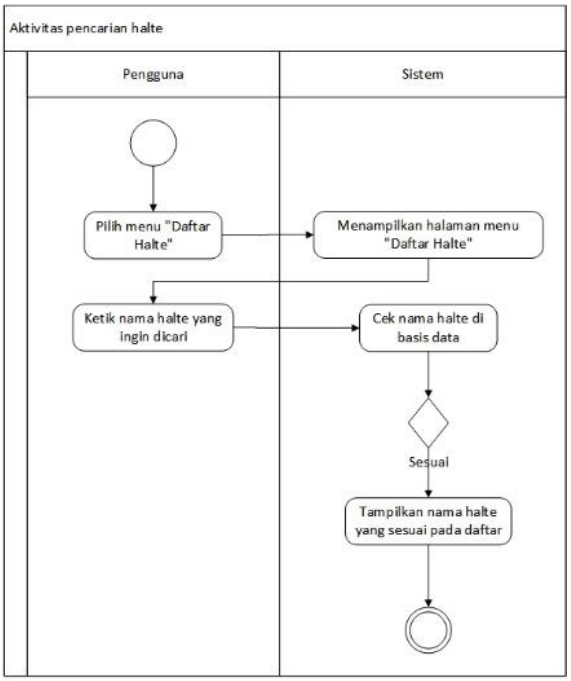
Diagram aktivitas menggambarkan aliran proses suatu perilaku atau aktivitas entitas yang ada di dalam sistem. Entitas dalam diagram aktivitas dapat berupa pengguna atau pun sistem itu sendiri.

Diagram aktivitas kegiatan inti pada aplikasi sistem presensi dapat dilihat pada Gambar 4. Kegiatan inti pada aplikasi .BRT merupakan pencarian rute yang perlu ditempuh oleh pengguna.



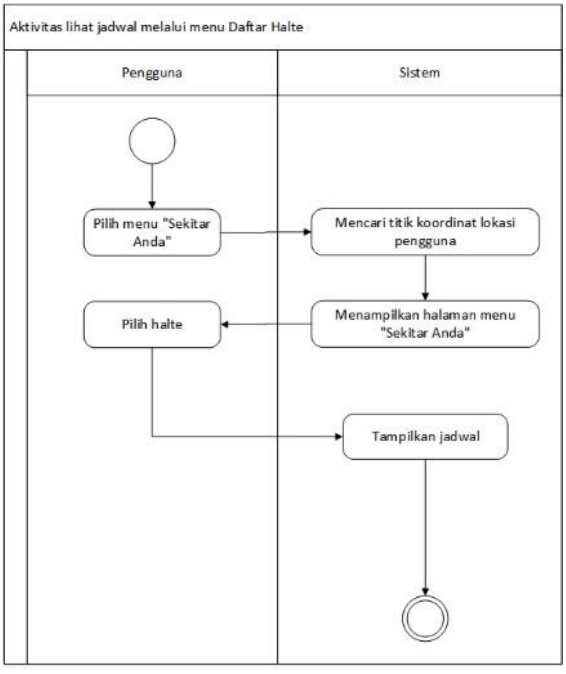
Gambar 4. Diagram aktivitas kegiatan inti

Pada Gambar 4 dapat dilihat diagram aktivitas pengguna saat melakukan aktivitas pencarian rute. Setelah pengguna memasukkan halte asal dan halte tujuan, sistem akan mencari rute mana yang harus ditempuh pengguna lalu menampilkan daftar halte yang harus dilewati.



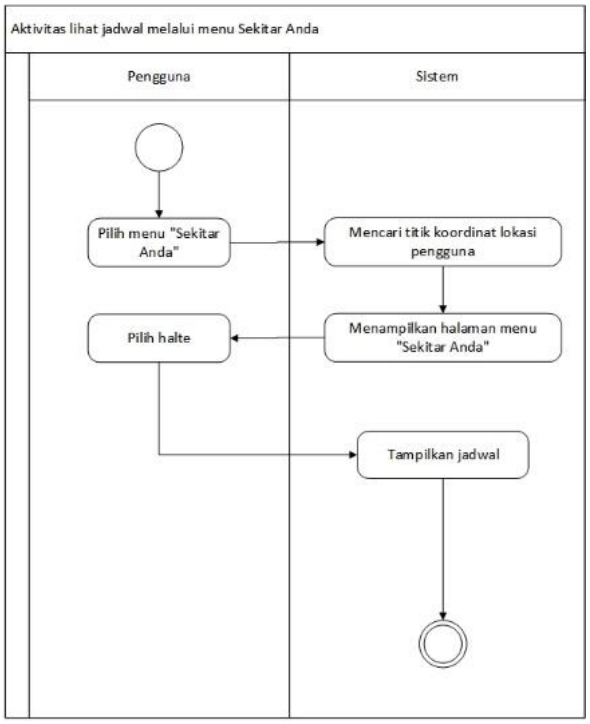
Gambar 5. Diagram aktivitas pencarian halte

Pada Gambar 5 dapat dilihat diagram aktivitas pengguna dapat mencari halte dari menu “Daftar Halte”. Pengguna dapat memasukkan kata kunci pada kolom yang tersedia lalu sistem akan mensortir daftar sesuai kata kunci yang telah dimasukkan.



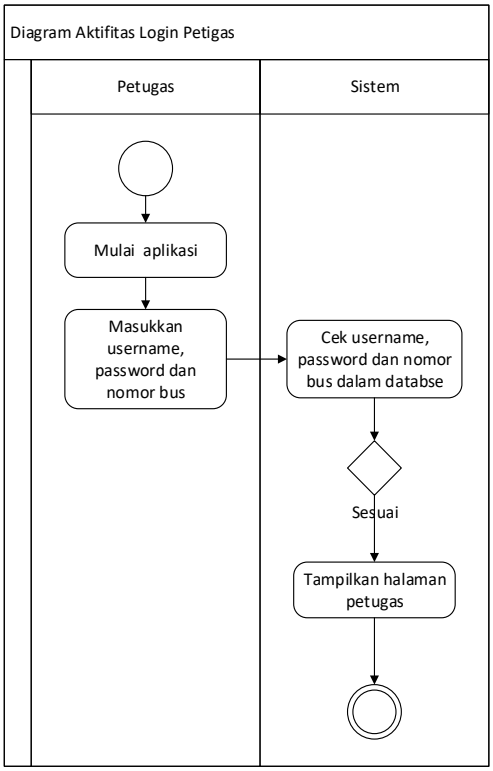
Gambar 6. Diagram aktivitas lihat jadwal melalui menu daftar halte

Pada Gambar 6 dapat dilihat diagram aktivitas pengguna saat melakukan aktivitas lihat jadwal melalui menu “Daftar Halte”. Pengguna dapat melihat jadwal bus yang sedang beroperasi melalui menu “Daftar Halte” dengan memilih halte pada daftar yang sudah tersedia.



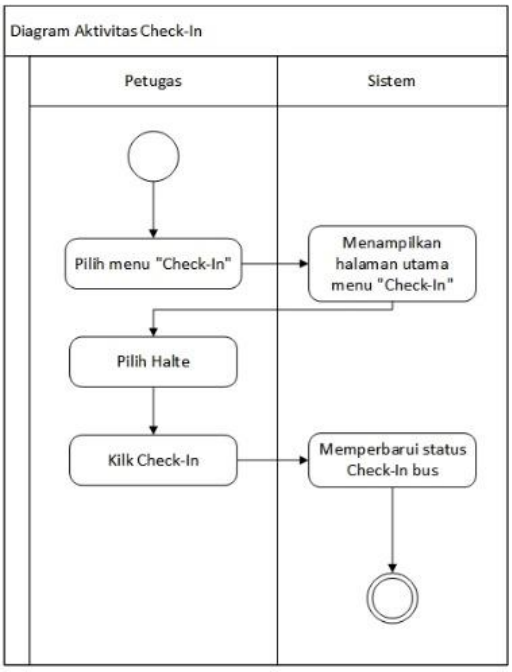
Gambar 7. Diagram aktivitas lihat jadwal melalui menu sekitar anda

Pada Gambar 7 dapat dilihat diagram aktivitas pengguna saat melakukan aktivitas lihat jadwal melalui menu “Sekitar Anda”. Pengguna dapat melihat jadwal bus yang sedang beroperasi melalui menu “Sekitar Anda” dengan memilih *marker* halte yang terdapat pada peta.



Gambar 8. Diagram aktivitas login petugas

Pada Gambar 8 dapat dilihat diagram aktivitas petugas saat melakukan *login*. Setelah petugas berhasil melakukan *login*, petugas dapat melakukan *check-in* pada halte yang sedang dilewati.

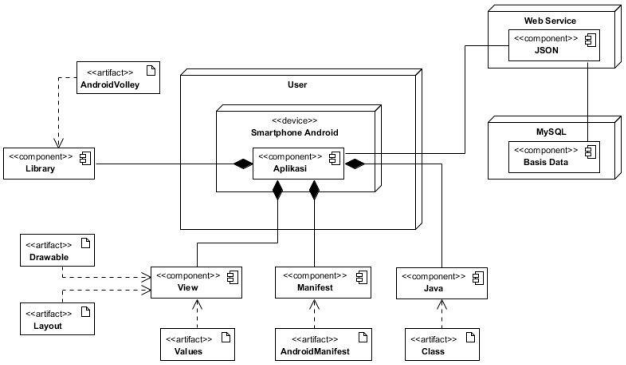


Gambar 9. Diagram aktivitas check-in halte

Pada Gambar 9 dapat dilihat diagram aktivitas petugas saat melakukan *check-in*. Petugas memilih halte yang sedang dilewati lalu menekan tombol *check-in* untuk menambahkan atau memperbarui jadwal bus pada sisi pengguna.

2.3.3. Diagram *Deployment*

Diagram *deployment* menggambarkan arsitektur sistem yang dapat berupa konfigurasi komponen-komponen perangkat keras, atau konfigurasi komponen-komponen perangkat lunak. Diagram *deployment* aplikasi sistem presensi dapat dilihat pada Gambar 9.



Gambar 8. Diagram *deployment* aplikasi .BRT

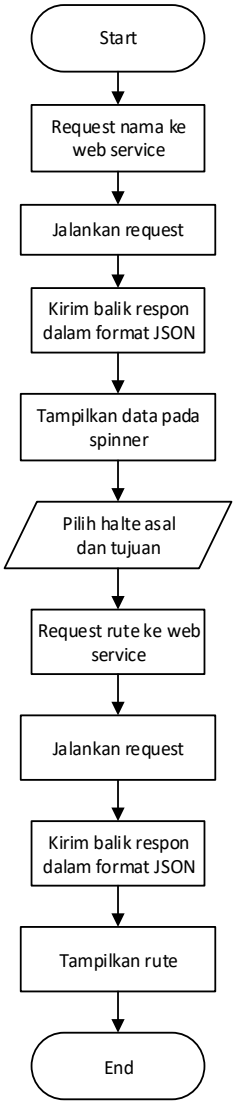
Secara garis besar, sistem terdiri dari *user*, *web service*, dan *server*. *User* menggunakan fitur REST(*Representation State Trasnfer)* pada *web sercive* untuk melakukan *request* data dari *user* ke basis data MySQL melalui API(*Application Program Interface*). Data kemudian dikirimkan kembali ke sisi *user* berupa data *array* dengan format JSON sehingga dapat diakses oleh *user* dan dapat ditampilkan pada aplikasi android.

3. Hasil dan Analisa

3.1. Implementasi Sistem

3.1.1. Implementasi Aplikasi .BRT

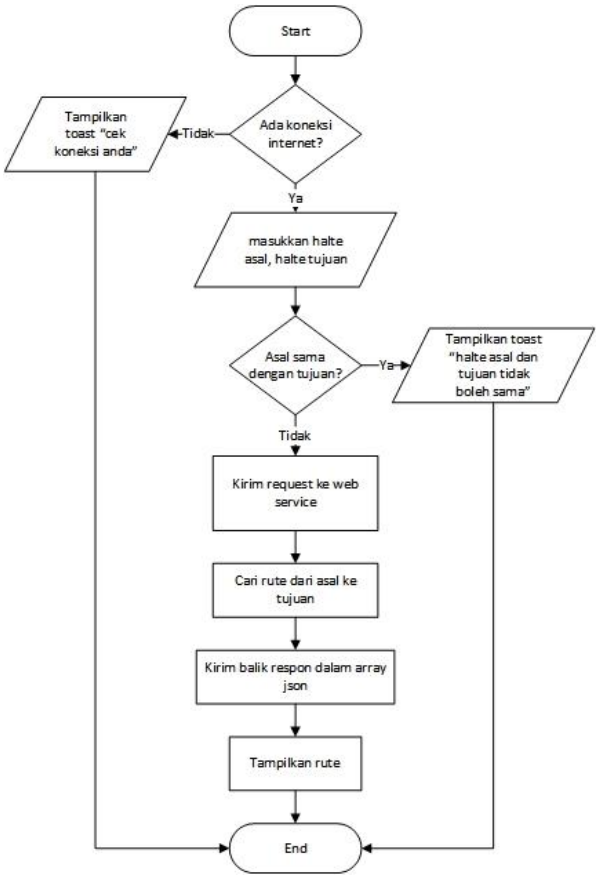
Berdasarkan hasil analisa sistem, penerapan algoritma Dijkstra pada aplikasi .BRT adalah dalam pencarian rute. Pada saat memilih menu “Cari Rute”, sistem akan mengambil data halte dari basis data untuk ditampilkan pada *spinner*. Pengguna dapat memilih halte asal dan tujuan dari dua *spinner* yang telah disediakan. Setelah memilih halte asal dan tujuan, pengguna dapat menekan tombol “Cari Rute” untuk mendapatkan rute yang perlu ditempuh. Sistem akan mengirim *request* ke *web service* untuk mendapatkan rute. *Web service* akan mencari id halte yang telah dipilih oleh pengguna pada tabel list_simpul. Setelah itu *web service* akan mencari rute dengan jumlah *hop* paling sedikit. Respon lalu akan dikirim dalam format JSON agar dapat ditampilkan pada aplikasi .BRT. Alur kerja penerapan algoritma Dijkstra dalam proses pencarian rute dapat dilihat melalui diagram alur pada Gambar 11.



Gambar 11. Alur kerja proses pencarian rute

Tahap selanjutnya adalah tahap implementasi ketika analisa sistem diterapkan ke bentuk pengkodean, sehingga dapat diketahui apakah sistem telah dibuat sudah sesuai dengan perancangan. Salah satu fitur inti program terdapat pada berkas get_rute.php.

Pada fitur pencarian rute, *query* yang digunakan untuk mengakses basis data berfungsi untuk mencari halte yang harus dilewati pengguna untuk sampai ke halte tujuan. Respon yang dihasilkan oleh *query* berupa *array* JSON yang kemudian ditampilkan pada aplikasi .BRT. Gmabar 12 menunjukkan proses aliran data dari proses *rquest* hingga dapat menampilkan rute yang harus diambil oleh pengguna.

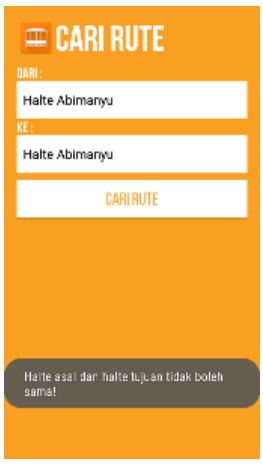


Gambar 12. Proses aliran data pencarian rute

Berdasarkan Gambar 12 dapat diketahui bahwa respon yang akan muncul apabila kondisi pertama terpenuhi adalah “Gagal mengambil data, cek koneksi internet”. Hal ini mengakibatkan aplikasi .BRT menampilkan respon *toast* seperti pada Gambar 13.

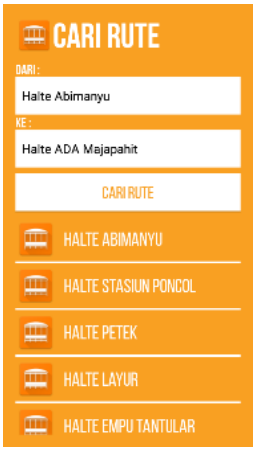
Gambar 13. Pencarian rute dengan respon “Gagal mengambil data, cek koneksi internet”

Apaila kondisi pertama tidak terpenuhi maka sistem akan mencari kondisi berdasarkan *query* yang kedua. Apabila kondisi kedua terpenuhi, maka respon yang akan muncul adalah "halte asal dan halte tujuan tidak boleh sama". Hal ini mengakibatkan aplikasi .BRT menampilkan respon *toast* seperti pada Gambar 14.



Gambar 13. Pencarian rute dengan respon "Halte asal dan tujuan tidak boleh sama"

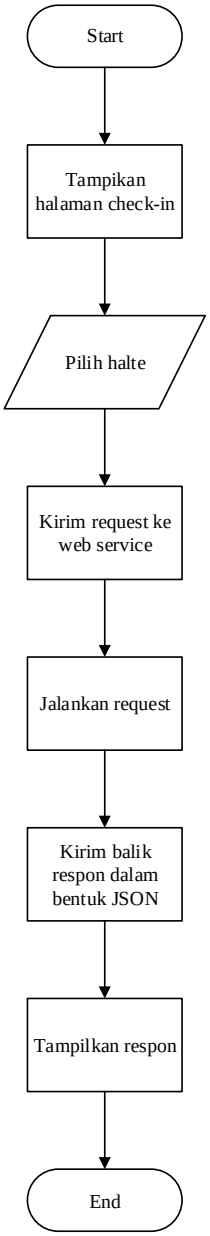
Apaila kondisi kedua tidak terpenuhi maka sistem akan mencari kondisi berdasarkan *query* yang ketiga. Apabila kondisi ketiga terpenuhi, maka respon yang akan muncul adalah *array* JSON yang kemudian akan ditampilkan dalam bentuk daftar. Hal ini mengakibatkan aplikasi .BRT menampilkan daftar rute yang perlu ditempuh pengugna seperti pada Gambar 15.



Gambar 15. Pencarian rute dengan respon daftar rute

3.1.2. Implementasi Aplikasi .PETUGAS

Berdasarkan hasil analisa sistem, aplikasi .PETUGAS menggunakan *library* AndroidVolley untuk melakukan *check-in* pada halte. Ketika petugas memasuki halaman *check-in*, sistem akan mengambil data halte pada bassi data untuk ditampilkan pada *spinner*. Petugas dapat melakukan *check-in* dengan memilih halte yang sedang disinggahi lalu menekan tombol "Check-in". Sistem selanjutnya akan mengirimkan *request* ke *web service*. *Web service* akan mencari data bus pada basis data lalu memproses *request* tesebut. Respon akan dikirim balik damalm format JSON agar dapat ditampilkan pada aplikasi .PETUGAS. Alur kerja *check-in* pada halte dapat dilihat pada Gambar 16.

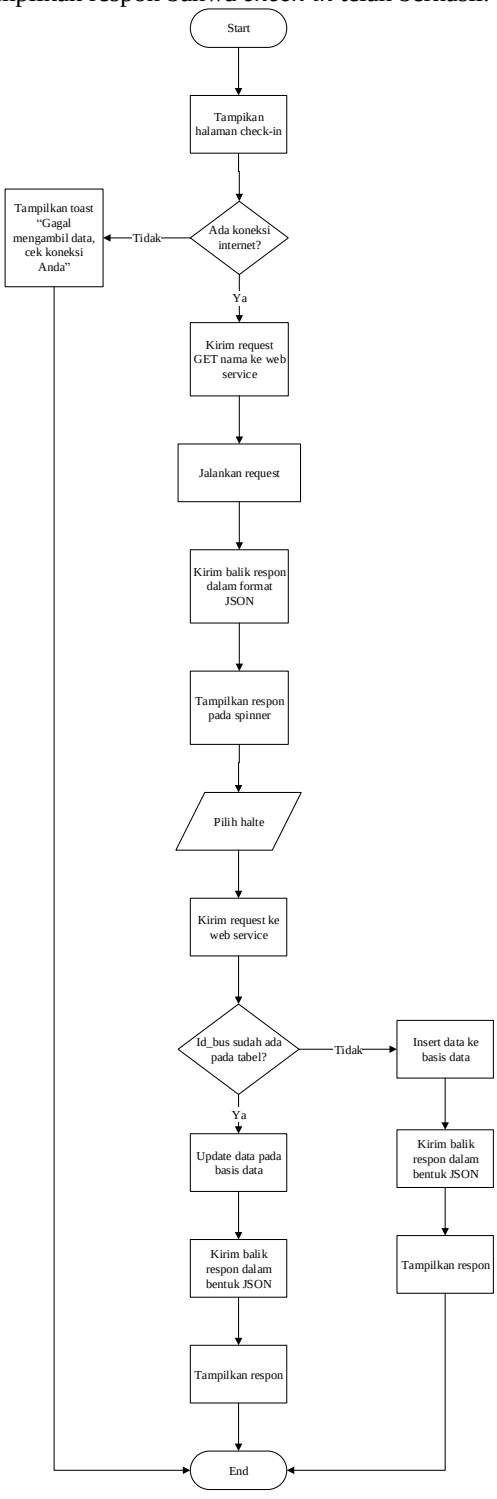


Gambar 16. Alur kerja proses check-in pada halte

Tahap selanjutnya adalah tahap implementasi ketika analisa sistem diterapkan ke bentuk pengkodean, sehingga dapat diketahui apakah sistem telah dibuat sudah sesuai dengan perancangan. Salah satu fitur inti program terdapat pada berkas checkin.php.

Pada fitur pencarian rute, *query* yang digunakan untuk mengakses basis data berfungsi untuk melakukan *check-*

in pada halted an memperbarui jadwal bus yang sedang beroperasi. Respon yang dihasilkan oleh *query* berupa respon dengan format JSON yang kemudian ditampilkan pada aplikasi .PETUGAS. Gmabar 17 menunjukkan proses aliran data dari proses *rquest* hingga dapat menampilkan respon bahwa *check-in* telah berhasil.



Gambar 17. Proses aliran data check-in pada halte

Berdasarkan Gambar 17 dapat diketahui bahwa respon yang akan muncul apabila kondisi pertama terpenuhi adalah "Gagal mengambil data, cek koneksi Anda". Hal ini mengakibatkan aplikasi .PETUGAS menampilkan respon *toast* seperti pada Gambar 18.

Gambar 18. Check-in halte dengan respon "Gagal mengambil data, cek koneksi Anda"

Apaila kondisi pertama tidak terpenuhi maka sistem akan mencari kondisi berdasarkan *query* yang kedua. Apabila kondisi kedua terpenuhi, maka respon yang akan muncul adalah "(nama halte) check-in berhasil". Hal ini mengakibatkan aplikasi .PETUGAS menampilkan respon *toast* seperti pada Gambar 19.

Gambar 19. Check-in halte dengan respon "Halte Abimanyu chek-in berhasil"

Apaila kondisi kedua tidak terpenuhi maka sistem akan mencari kondisi berdasarkan *query* yang ketiga. Apabila kondisi ketiga terpenuhi, maka respon yang akan muncul adalah "(nama halte) check-in berhasil". Hal ini mengakibatkan aplikasi .PETUGAS menampilkan respon *toast* seperti pada Gambar 20/

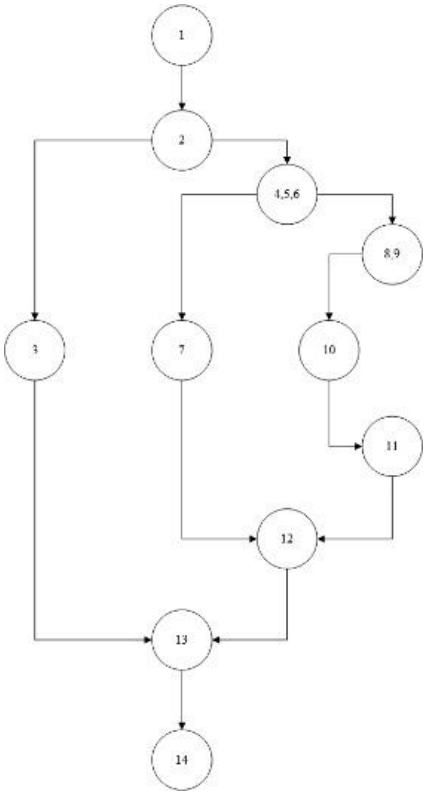
Gambar 20. Check-in halte dengan respon "Halte Abimanyu chek-in berhasil"

3.2. Pengujian Sistem
3.2.1. Pengujian *White Box*
3.2.1.1. Pengujian *White Box* Aplikasi .BRT

Pengujian *white box* digunakan untuk meyakinkan semua perintah dan kondisi pada fitur utama aplikasi dieksekusi secara minimal. Pengujian *white box* menggunakan dua alat yaitu *flow graph* yang digunakan untuk menggambarkan alur dari algoritma dan *graph matrix* yang digunakan untuk menggenerasi *flow graph*. Pengujian ini dilakukan pada fitur pencarian rute untuk mengetahui apakah semua perintah pada proses pencarian rute berjalan dengan baik. Berikut merupakan *pseudocode* yang digunakan.

```
1.  GET request $id dari web service
2.  If error request data
3.  Then echo "Gagal mengambil data, cek
    koneksi Anda"
4.  Else
5.  Tampilkan data pada spinner
6.  If $asal = $tujuan
7.  Then echo "Halte asal dan halte tujuan
    tidak boleh sama"
8.  Else
9.  If $asal != $tujuan
10. Then echo (json_encode('arrayrute'))
11. End if
12. End if
13. End if
14. Show response
```

Setelah membuat *pseudocode* sebagai algoritma proses pengambilan data, dilakukan pengubahan menjadi *flow* graph. Pengubahan *pseudocode* menjadi *flow graph* dapat dilihat pada Gambar 16.



Gambar 21. *Flow graph* hasil pengubahan dari *pseudocode*

Pada Gambar 21 menggambarkan *flow graph* dengan lingkaran menggambarkan kondisi, sedangkan anak panah menggambarkan aksi. Dari Gambar 16 dapat dihitung *cyclomatic complexity* sebagai berikut:

$$V(G) = E - N + 2$$
$$V(G) = 12 - 11 + 2 = 3$$

Dimana E merupakan jumlah *edge* atau anak panah pada grafik alir, sedangkan N merupakan *node* atau lingkaran pada grafik alir. Dari perhitungan rumus tersebut didapatkan *cylomatic complexity* bernilai 3. Berdasarkan hasil tersebut, maka dibutuhkan minimal 3 *path* untuk melewati semua *node*. Tabel 1 merupakan tabel berisi *path cylomatic complexity*.

Tabel 1. *Path cylomatic complexity*

Path	Aliran Proses
Path 1	1,2,3,13,14
Path 2	1,2,4,5,6,7,12,13,14
Path 3	1,2,4,5,6,8,9,10,11,12,13,14

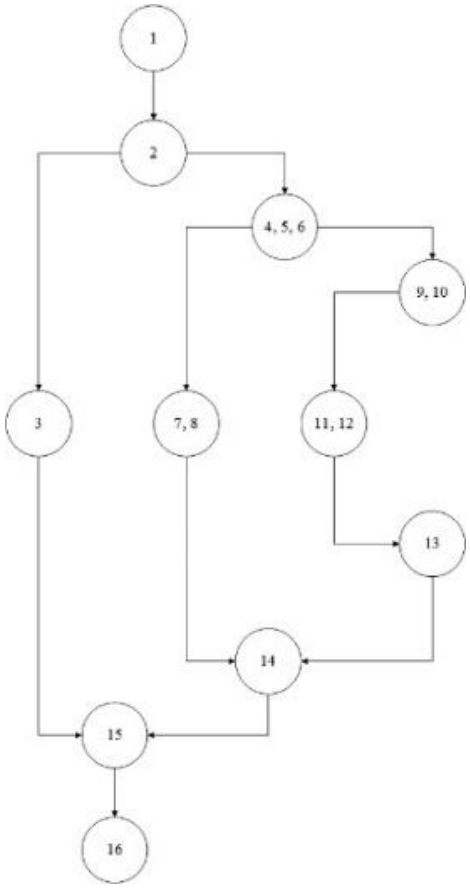
Hasil pengujian *white-box* menunjukkan semua alur pengujian terlewati.

3.2.1.2. Pengujian *White Box* Aplikasi .PETUGAS

Pengujian *white box* digunakan untuk meyakinkan semua perintah dan kondisi pada fitur utama aplikasi dieksekusi secara minimal. Pengujian *white box* menggunakan dua alat yaitu *flow graph* yang digunakan untuk menggambarkan alur dari algoritma dan *graph matrix* yang digunakan untuk menggenerasi *flow graph*. Pengujian ini dilakukan pada fitur *check-in* halte untuk mengetahui apakah semua perintah pada proses *check-in halte*berjalan dengan baik. Berikut merupakan *pseudocode* yang digunakan.

```
1.  POST request $nama dan $id dari web
    service
2.  If erro rewuest data
3.  Then Toast "Gagal mengambil data, cek
    koneksi Anda"
4.  Else
5.  Tampilkan data pada spinner
6.  If $id_bus not exist
7.  Then INSERT into jadwal
8.  Echo $nama "check in berhasil"
9.  Else
10. If $id_bus exist
11. Then UPDATE whre $id=$id
12. Echo $nama "check in berhasil"
13. End if
14. End if
15. End if
16. Show response
```

Setelah membuat *pseudocode* sebagai algoritma proses pengambilan data, dilakukan pengubahan menjadi *flow* graph. Pengubahan *pseudocode* menjadi *flow graph* dapat dilihat pada Gambar 17.



Gambar 17. *Flow graph* hasil pengubahan dari *pseudocode*

Pada Gambar 17 menggambarkan *flow graph* dengan lingkaran menggambarkan kondisi, sedangkan anak panah menggambarkan aksi. Dari Gambar 16 dapat dihitung *cyclomatic complexity* sebagai berikut:

$$V(G) = E - N + 2$$
$$V(G) = 12 - 11 + 2 = 3$$

Dimana E merupakan jumlah *edge* atau anak panah pada grafik alir, sedangkan N merupakan *node* atau lingkaran pada grafik alir. Dari perhitungan rumus tersebut didapatkan *cylomatic complexity* bernilai 3. Berdasarkan hasil tersebut, maka dibutuhkan minimal 3 *path* untuk melewati semua *node*. Tabel 2 merupakan tabel berisi *path cylomatic complexity*.

Tabel 2. *Path cylomatic complexity*

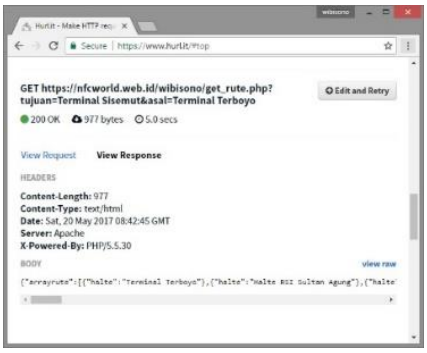
Path	Aliran Proses
Path 1	1,2,3,15,16
Path 2	1,2,4,5,6,7,8,14,15,16
Path 3	1,2,4,5,6,9,10,11,12,13,14.15,16

Hasil pengujian *white-box* menunjukkan semua alur pengujian terlewati.

3.2.2. Pengujian Komunikasi Data

3.2.2.1. Pengujian Komunikasi Pencarian Rute

Pengujian komunikasi data merupakan pengujian untuk mengetahui hasil dari pertukaran data yang dilakukan *web service* berupa besar data yang dikirimkan dan durasi waktu pengiriman data. Gambar 18 adalah salah satu hasil pengujian komunikasi data dengan berkas get_rute.php



Gambar 18. Hasil pengujian komunikasi data pada berkas get_rute.php

Pada Gambar 18 terlihat bahwa pengujian komunikasi data telah menangkap balikan data yang dilakukan *web service* dengan durasi waktu menyelesaikan proses eksekusi pengujian adalah 5000 ms, serta besar data yang diterima berukuran 977 bytes. Setelah melakukan lima kali uji coba pada berkas get_rute.php, maka diperoleh hasil seperti pada Tabel 3.

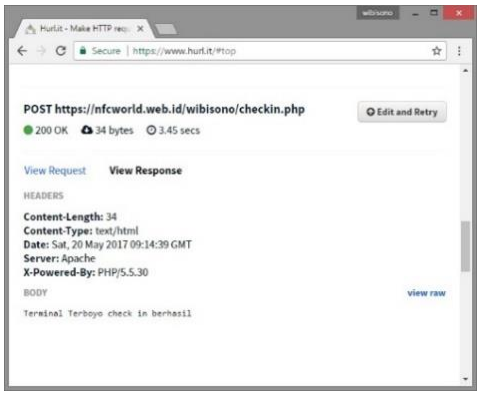
Tabel 3. Hasil pengujian komunikasi data pada berkas get_rute.php

Uji ke-	Respon	Besar Data (bytes)	Waktu Durasi (ms)
1	JSON iarray	977	5000
2	JSON iarray	996	6600
3	JSON iarray	1120	4780
4	JSON iarray	1029	5350
5	JSON iarray	287	4480
TOTAL		**4400**	**26210**
RATA-RATA		**880**	**5242**

Pada Tabel 3 dapat dilihat bahwa setelah dilakukan lima kali uji coba, diperoleh rata-rata data yang diterima sebesar 880 bytes dengan rata-rata durasi waktu sebesar 5242 ms.

3.2.2.2. Pengujian Komunikasi *Check-In*

Pengujian komunikasi data merupakan pengujian untuk mengetahui hasil dari pertukaran data yang dilakukan *web service* berupa besar data yang dikirimkan dan durasi waktu pengiriman data. Gambar 19 adalah salah satu hasil pengujian komunikasi data dengan berkas checkin.php



Gambar 19. Hasil pengujian komunikasi data pada berkas checkin.php

Pada Gambar 19 terlihat bahwa pengujian komunikasi data telah menangkap balikan data yang dilakukan *web service* dengan durasi waktu menyelesaikan proses eksekusi pengujian adalah 3450 ms, serta besar data yang diterima berukuran 34 bytes. Setelah melakukan lima kali uji coba pada berkas checkin.php, maka diperoleh hasil seperti pada Tabel 4.

Tabel 4. Hasil pengujian komunikasi data pada berkas checkin.php

Uji ke-	Respon	Besar Data (bytes)	Waktu Durasi (ms)
1	Terminal Terboyo check-in berhasil	34	3450
2	Terminal Penggaron check-in berhasil	36	3260
3	Halte Balai Kota check-in berhasil	34	4530
4	Halte Akpol check-in berhasil	29	5250
5	Terminal Terboyo check-in berhasil	34	4920
TOTAL		**167**	**21410**
RATA-RATA		**33,4**	**4280**

Pada Tabel 4 dapat dilihat bahwa setelah dilakukan lima kali uji coba, diperoleh rata-rata data yang diterima sebesar 33,4 bytes dengan rata-rata durasi waktu sebesar 4280 ms.

3.2.3. Pengujian Pada Perangkat Keras

Pengujian pada perangkat keras merupakan pengujian secara langsung kepada beberapa perangkat. Pengujian perangkat keras bertujuan untuk mengetahui kompatibilitas aplikasi .BRT dan .PETUGAS pada beberapa merek dan spesifikasi *smartphone*. Setelah melakukan pengujian, maka didapatkan hasil dan analisa pada Tabel 5.

Tabel 5. Hasil pengujian pada perangkat keras.

Merk	Spesifikasi	Keterangan
Lenovo A2010	- Resolusi Layar : 480 x 854 pixels - Dimensi Layar : 4,5 inci - Sistem Operasi : Android versi 5.1 (Lollipop) - Memori : 8 GB, 1.5GB RAM - Processor : Quad-core 1.0 GHz Cortex-A53	Aplikasi dapat berjalan dengan baik dan lancer, sedikit sekaliruang kosong tersisa pada tampilan
ASUS Zenfone Selfie	- Resolusi Layar : 1080 x 1920 pixels - Dimensi Layar : 5,5 inci - Sistem Operasi : Android versi 6.0 (Marshmallow) - Memori : 32 GB, 3 GB RAM - Processor : 4x1.7 GHz Cortex-A53 & 4x1.0 GHz Cortex-A5	Aplikasi dapat berjalan dengan baik dan lancar, sedikit ruang kosong tersisa pada tampilan
Samsung A5 (2017)	- Resolusi Layar : 1080 x 1920 pixels - Dimensi Layar : 5,5 inci - Sistem Operasi : Android versi 6.0.1 (Marshmallow) - Memori : 32 GB, 3GB RAM - Processor : Octa-core 1.9 GHz Cortex-A53	Aplikasi dapat berjalan dengan baik dan lancar, banyak ruang kosong tersisa pada tampilan
Lenovo A6010 Plus	- Resolusi Layar : 720 x 1280 pixels - Dimensi Layar : 5 inci - Sistem Operasi : Android versi 5.0 (Lollipop) - Memori : 16 GB, 2GB RAM - Processor : Quad-core 1.2 GHz Cortex-A53	Aplikasi dapat berjalan dengan baik dan lancar, sedikit ruang tersisa pada tampilan
Lenovo K4 Note	- Resolusi Layar : 1080 x 1920 pixels - Dimensi Layar : 5,5 inci - Sistem Operasi : Android versi 5.1.1 (Lollipop) - Memori : 16 GB, 3 GB RAM - Processor : Octa-core 1.3 GHz Cortex-A53	Aplikasi dapat berjalan dengan baik dan lancar, sedikit ruang kosong tersisa pada tampilan.
Samsung Galaxy Note 3	- Resolusi Layar : 720 x 1280 pixels - Dimensi Layar : 5,5 inci - Sistem Operasi : Android versi 5.5.1 (Lollipop) - Memori : 16 GB, 2 GB RAM - Processor : Hexa-core (4x1.3 GHz Cortex A7 & 2x1.7 GHz Cortex A15)	Aplikasi dapat berjalan dengan baik dan lancar, sedikit ruang kosong tersisa pada tampilan
Samsung Galaxy J5 Prime	- Resolusi Layar : 720 x 1280 pixels - Dimensi Layar : 5 inci - Sistem Operasi : Android versi 6.0.1 (Marshmallow) - Memori : 16 GB, 2 GB RAM - Processor : Quad-core 1.4 GHz Cortex-A53	Aplikasi dapat berjalan dengan baik dan lancar, sedikit ruang kosong tersisa pada tampilan

Berdasarkan Tabel 5 terlihat bahwa pengujian kompatibilitas perangkat keras, aplikasi ini dapat di implementasikan pada berbagai tipe perangkat keras. Hal ini sesuai dengan spesifikasi minimum aplikasi .BRT dan .PETUGAS yang membutuhkan perangkat dengan sistem operasi Android versi 4.2 (Jelly Bean) atau lebih baru. Perbedaan proporsi tampilan pada berbagai perangkat yang telah diujui disebabkan oleh perbedaan ukuran layar dan kepadatan pixel yang dimiliki masing-masing perangkat keras.

4. Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dapat ditarik kesimpulan bahwa selain dapat mencari rute bus yang perlu ditempuh pengguna, aplikasi ini juga dapat menampilkan halte di sekitar pengguna, menampilkan daftar halte dan menampilkan jadwal bus yang sedang beroperasi. Aplikasi ini telah berhasil diimplementasikan pada beberapa perangkat keras dengan merek dan spesifikasi yang berbeda. Sistem mampu melakukan pencarian rute dengan rata-rata durasi waktu sebesar 5242 ms dan rata-rata ukuran respon sebesar 880 bytes. Pada penelitian selanjutnya, penulis berharap pengenmbang dapat menerapkan Google Places API pada sistem ini dan membuat fitur jadwal pada halte khusus untuk koridor halte tersebut.

Referensi

[1] E. W. Dijkstra, “A Note on Two Probles in Connexion with Graphs,” *Numer. Math.*, vol. 1, no. 1, pp. 269–271, 1959.

[2] I. W. G. S. Wijaya and E. H. Susanto, “Penerapan Algoritma Dijkstra Untuk Menemukan Rute,” pp. 1–8, 2012.

[3] S. Vivas and G. Riaño, “Mobile device programming for finding optimal path on a fast-bus system,” *Proc. 2006 IEEE Syst. Inf. Eng. Des. Symp. SIEDS’06*, pp. 196–201, 2007.

[4] R. I. Nugraha and F. N. Hakim, “Rancang Bangun Aplikasi Informasi Lokasi Shelter Bus Trans,” *J. Teknol. Inf. dan Komun.*, vol. 6 Nomor, pp. 9–14, 2015.

[5] D. Ardana and R. Saputra, “Penerapan Algoritma Dijkstra pada Aplikasi Pencarian Rute Bus Trans Semarang,” no. Snik, pp. 299–306, 2016.