Jurnal Teknologi Informasi dan Ilmu Komputer (JTIIK) DOI: 10.25126/jtiik.202071680
Vol. 7, No. 4, Agustus 2020, hlm. 653-660 p-ISSN: 2355-7699
Akreditasi KEMENRISTEKDIKTI, No. 30/E/KPT/2018 e-ISSN: 2528-6579

APLIKASI PENENTUAN RUTE DAN WAKTU TEMPUH KE HALTE TRANSJAKARTA TERDEKAT DENGAN ALGORITME DIJKSTRA BERBASIS LOCATION BASE SYSTEM

Ra'idah Naufaliana Dewi[1], Dwi Atmodjo W.P.[2], Mardiana Purwaningsih*[3]

[1,2,3]Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Perbanas Institute
Email: rnaufalianadewi@gmail.com[1], dwiatmodjo@gmail.com[2], mardiana@perbanas.id[3]
*Penulis Korespondensi

(Naskah masuk: 21 Januari 2019, diterima untuk diterbitkan: 6 Maret 2019)

Abstrak

Saat ini sudah ada aplikasi yang dapat diinstal di ponsel pintar untuk membantu mencari posisi halte terdekat, akan tetapi pengguna harus benar-benar tahu titik keberadaannya saat itu. Hal ini dapat menyulitkan apabila penumpang tersebut adalah pendatang yang tidak mengetahui dengan benar posisinya. Penelitian dengan tujuan menentukan rute dan waktu tempuh ke halte Transjakarta terdekat ini dibuat dengan pendekatan algoritme Dijkstra pada aplikasi mobile di atas platform Android. Aplikasi ini akan dibantu oleh GPS untuk mencari halte terdekat dari posisi pengguna. Aplikasi ini juga dilengkapi dengan deskripsi perjalanan, estimasi waktu tempuh, dan rute alternatif menuju halte Transjakarta. Kelebihan aplikasi ini adalah apabila Wi-Fi atau data seluler tidak berfungsi tetap dapat digunakan karena metode ini akan tetap menunjukkan rute yang perlu ditempuh dengan deskripsi perjalanan yang dimilikinya. Penentuan rute terdekat dalam aplikasi ini tidak perlu menentukan titik akhir seperti halnya aplikasi yang menggunakan algoritme Djikstra lainnya, bahkan dapat memberikan beberapa alternatif halte terdekat. Hal ini dimungkinkan karena aplikasi ini juga menggunakan algoritme Bubblesort untuk mengurutkan halte. Karena tiap halte Transjakarta sudah terintegrasi sehingga kemana pun tujuan dapat diakses dari halte terdekat dan di halte tersebut sudah ada informasi halte transit sesuai dengan tujuan.

Kata kunci: *pencarian rute terdekat, waktu tempuh, Algoritme Dijkstra*

APPLICATION OF ROUTING AND TIME DETERMINATION TO THE NEAREST TRANSJAKARTA BUS STOP WITH DIJKSTRA ALGORITHM BASED ON LOCATION BASE SYSTEM

Abstract

Currently there are applications that can be installed on a smart phone to help find the position of the nearest bus stop, but the user must really know the point of existence at that time. This can be difficult if the passenger is a migrant who does not know his position correctly. The approach used to determine the route and travel time to the nearest Transjakarta bus stop in this study is to use Dijkstra's algorithm on a mobile application on the Android platform. This application will be assisted by GPS to find the nearest stop from the user's position. This application is also equipped with a description of the trip, estimated travel time, and alternative routes to the Transjakarta bus stop. The advantage of this application is that if Wi-Fi or cellular data does not work it can still be used because this method will still show the route that needs to be taken with the description of the trip it has. Determining the closest route in this application does not need to determine the end point as well as applications that use other Djikstra algorithms, can even provide several alternative stops nearby. This is possible because this application also uses the Bubblesort algorithm to sort stops. Because each Transjakarta bus stop has been integrated so that wherever the destination can be accessed from the nearest bus stop and at the stop there is already a transit stop information in accordance with the destination.

Keywords: *nearest bus stopped, time determination, Dijkstra Algorithm*

1. PENDAHULUAN

Transjakarta merupakan moda transportasi massal pendukung aktivitas penduduk kota Jakarta. Transportasi ini hanya berhenti pada halte-halte tertentu sehingga calon penumpang harus menunggu di halte sebelum menaiki Transjakarta. Jumlah koridor dan halte yang tersedia pun terbilang banyak bahkan sampai puluhan yang menyebabkan calon penumpang harus tepat memilih halte sesuai dengan tujuan. Dengan kesibukan yang dimilikinya, calon penumpang membutuhkan informasi yang cepat

mengenai posisi dan rute tersebut. Biasanya mereka akan bertanya pada penduduk sekitar untuk menunjukkan rute terdekat menuju halte Transjakarta. Namun, hal ini menjadi tidak efisien mengingat bahwa tidak semua penduduk mengetahui informasi yang sama berdasarkan pertanyaan yang diajukan.

Calon penumpang juga mengandalkan internet untuk mendapat informasi mengenai halte Transjakarta terdekat dan rute terpendek menuju halte tersebut. Sebagai contoh saat memasukkan kata kunci "halte-TransJakarta-terdekat" pada kolom *search engine* maka seluruh informasi yang berkaitan dengan kata "halte", "TransJakarta", dan "terdekat" akan muncul di laman pencarian. Namun, metode ini memiliki kelemahan yaitu informasi yang sama tidak akan muncul pada kata kunci yang berbeda walaupun memiliki tujuan yang sama seperti halnya ketika memasukkan kata kunci "halte-busway-terdekat". Jumlah informasi dari kedua kata kunci itu pun tidak akan sama walaupun maksud dan tujuannya sama yaitu untuk mendapatkan rute terdekat menuju halte Transjakarta. Masalah lain yang muncul ketika posisi halte terdekat sudah diketahui adalah calon penumpang Transjakarta perlu tahu jarak dan perkiraan waktu tempuh menuju halte tersebut.

Penelitian untuk mencari jalur terpendek suatu rute atau lokasi tertentu sudah banyak dilakukan baik yang berbasis web maupun aplikasi, dan memungkinkan untuk menggunakan berbagai algoritme. Beberapa topik penelitian yang sudah memanfaatkan algoritme untuk pencarian jalur terpendek banyak digunakan pada bidang pariwisata. Dengan metode ini maka para wisatawan akan mendapatkan informasi lokasi wisata sehingga alokasi waktu untuk berwisata menjadi lebih efisien (Gunawan, 2015; Gusmão, Pramono, & Sunaryo, 2013; Suryo Saputro, 2013). Pada penelitian yang dilakukan oleh Suryo Saputra (2013) aplikasi berbasis mobile tersebut diharapkan dapat membantu wisatawan di kota Manado dengan menampilkan rute antar tempat wisata dan hotel dari posisi pengguna. Paling banyak penelitian pencarian jalur terpendek ini digunakan pada optimalisasi di bidang lalu lintas (Dwi, Saputra & Ardana, 2016; Galán-García, Aguilera-Venegas, Galán-García, & Rodríguez-Cielos, 2015; Suryo Saputro, 2013), tetapi juga pencarian lokasi rumah sakit terdekat yang akan sangat membantu pengguna dalam kondisi darurat (Budihartanti & Pandiangan, 2016). Dengan demikian pada umumnya penelitian pada bidang ini ditujukan untuk memberikan informasi rute sekaligus petunjuk arah agar pengguna dapat sampai di lokasi yang diharapkan.

Aplikasi yang dirancang pada penelitian ini menggunakan pendekatan algoritme Djikstra. Algoritme ini digunakan dengan pertimbangan bahwa untuk pencarian jalur terpendek hasilnya lebih akurat (Chen, Shen, Zhou, & Yu, 2009; Rosyidi, Pradityo, Gunawan, & Sari, 2014; Singal Dcsa & Chhillar, 2014). Penelitian ini tidak melakukan pengujian beberapa algoritme yang biasanya digunakan untuk mencari rute terpendek, sehingga keputusan menggunakan algoritme Dikstra didasarkan pada hasil penelitian sebelumnya yang menyatakan bahwa Algoritme Dijkstra lebih intensif dalam komputasi untuk pencarian jalur optimum dalam suatu jaringan seperti internet. Waktu rata-rata eksekusi algoritme Djikstra juga lebih kecil dibanding algoritme Ant Colony, maka algoritme Djikstra banyak digunakan dalam pencarian jalur optimum pada jaringan internet dibanding algoritme lain. Hal ini disampikan dalam hasil penelitian Gusmão, Pramono, dan Sunaryo (2013) mengenai rancangan sistem informasi geografis pariwisata berbasis web.

Pada penelitian sebelumnya penggunaan algoritme Djikstra lebih banyak digunakan untuk menentukan rute terpendek dari dua buah lokasi, sehingga titik awal dan titik akhir sudah ditentukan di awal. Pada penelitian ini, untuk mencari informasi halte terdekat, maka yang ditentukan hanya titik awal saja dengan bantuan dari GPS atau lokasi pengguna saat ini, sedangkan titik akhir akan muncul pada pilihan di aplikasi. Titik akhir ini adalah pilihan halte yang dilengkapi dengan bobot jarak dan waktu tempuh. Hal ini memungkinkan untuk dilakukan karena penggunaan algoritme Djikstra ini juga dilengkapi dengan algoritme Bubble Sort, yang akan mengurutkan beberapa halte terdekat berdasarkan bobot jarak. Mengingat bahwa tiap halte Transjakarta sudah terintegrasi, termasuk informasi halte transit, maka tujuan atau lokasi yang akan dicapai oleh pengguna dapat dimulai pada halte terdekat mana pun.

Untuk pencarian jarak antar titik yang akan digunakan adalah Formula Haversine, karena formula ini lebih akurat dibandingkan dengan formula lain (Chopde & Nichat, 2013; Prasetyo & Hastuti, 2015). Hasil perhitungan dari formula Haversine ini akan diurutkan dengan algoritme Bubble Sort untuk membandingkan jarak yang diseleksi dengan jarak yang ditemukan berikutnya (Kumalasari, 2017; Rahayuningsih, 2016; Sonita & Nurtaneo, 2015). Aplikasi penentuan halte terdekat ini juga dilengkapi dengan jarak dan estimasi waktu tempuh, agar informasi bagi pengguna lebih lengkap. Kelebihan aplikasi ini adalah apabila tidak ada akses internet misal tidak ada *Wi-Fi* atau data seluler mati tetap dapat digunakan karena metode ini akan tetap menunjukkan rute yang perlu ditempuh dengan deskripsi perjalanan yang dimilikinya.

2. METODE PENELITIAN

2.1 Penentuan Halte Terdekat

Penentuan halte Transjakarta terdekat akan menggunakan pendekatan algoritme Dijkstra karena algoritme ini paling sering digunakan dalam mencari rute terpendek. Penggunaan algoritme ini cukup sederhana yaitu dengan menggunakan simpul-simpul sederhana pada jaringan jalan yang tidak kompleks. Algoritme Dijkstra ini dapat digunakan untuk mencari lintasan terpendek pada graf berarah maupun graf tak berarah. Gambar 1 memperlihatkan tahapan dalam menentukan lintasan halte Transjakarta terdekat pada penelitian ini.

1. Setiap halte disumsikan sebagai sebuah titik.
2. Memberi nilai atau bobot jarak untuk setiap titik dan berpindah ke titik lainnya. Titik awal diatur dengan nilai 0 dan terhadap titik lain diberikan nilai tak hingga (belum terisi).
3. Mengatur semua titik yang belum terlewati pada label sementara atau "*Open list*" dan mengatur titik awal sebagai "*Current distance*".
4. Dari titik keberangkatan dipertimbangkan melalui titik paling dekat yang belum terlewati dan dihitung jaraknya dari titik keberangkatan. Misal, jika titik keberangkatan halte A ke halte B memiliki bobot jarak 5 dan dari halte B ke halte C berjarak 4, maka jarak ke halte C melewati halte B menjadi 5+4=9. Jika jarak yg terekam sebelumnya lebih besar, maka data lama akan dihapus, dan jarak baru dengan nilai lebih kecil akan disimpan.
5. Langkah 4 akan diulang kembali sampai semua titik tetangga terlewati dan titik yang telah terlewati ditandai pada label permanen sebagai "*Closed list*". Jarak yang disimpan adalah jarak terakhir dengan bobot paling minimal, dan titik yang sudah terlewati tidak akan pernah dicek kembali.
6. Mengatur "*Open list*" dengan jarak terkecil (dari node keberangkatan) sebagai "*Current distance*" selajutnya dan dilanjutkan dengan kembali ke langkah 4.

Setelah tahapan penggunaan Algoritme Dijkstra, maka langkah selanjutnya dalam penelitian ini adalah menghitung jarak antar halte dengan menggunakan formula Haversine. Formula Haversine merupakan metode untuk mengetahui jarak antar dua titik yang ada di permukaan bumi, yang dalam hal ini adalah jarak antar halte Transjakarta (Chopde & Nichat, 2013; Prasetyo & Hastuti, 2015). Jarak antara 2 halte akan dihitung berdasarkan panjang garis lurus antara 2 halte pada garis bujur dan lintang. Langkah-langkah menggunakan rumus Haversine untuk menghitung jarak antar halte adalah sebagai berikut.

1. Memasukkan lalitude dan longitute titik koordinat keberangkatan dan tujuan.
2. Menghitung rumus Haversine (1).

$$d = 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right) \quad (1)$$

dimana:
d = jarak (km)
r = jari-jari bumi (6372.8 km)
ω = bujur timur (latitude)
λ = bujur barat (longitude)

Setelah jarak dari satu halte ke halte lain diperoleh, maka selanjutnya dilakukan pengurutan. Untuk pengurutan halte terdekat akan digunakan algoritme Bubble Sort. Langkah pada algoritme Bubble Sort adalah membandingkan jarak dari satu halte ke halte terdekat lainnya. Jika halte yang diseleksi jaraknya lebih jauh dari halte berikutnya maka posisi halte tersebut tersebut akan ditukar. Dengan model pengurutan seperti ini, posisi halte akan digeser satu per satu dari kanan ke kiri sesuai dengan jarak, dari angka yang paling kecil ke angka yang terbesar. Jika barisan bilangan disusun horizontal melainkan vertikal, maka terlihat seperti gelembung-gelembung (bubble) yang naik dari dasar akuarium. Oleh karena itu algoritme ini disebut algoritme Bubble Sort. Cara kerja algoritme Bubble Sort secara garis besar adalah melakukan pengulangan hasil perhitungan jarak halte secara terus menerus, menukar posisi halte apabila ditemukan jarak yang lebih kecil. Perbandingan jarak ini akan terus dilakukan sampai tidak ada lagi data yang dapat dipertukarkan (Kumalasari, 2017; Rahayuningsih, 2016; Sonita & Nurtaneo, 2015).

Selanjutnya adalah tahap *sorting* (pengurutan) yaitu proses pengurutan hasil perhitungan Haversine formula sebelumnya menjadi susunan data yang teratur. Prinsip yang diterapkan dalam algoritme Bubble Sort adalah mengurutkan tipe data array yang terdiri dari beberapa elemen dengan membandingkan elemen sekarang dengan elemen berikutnya. Langkah-langkahnya adalah sebagai berikut.

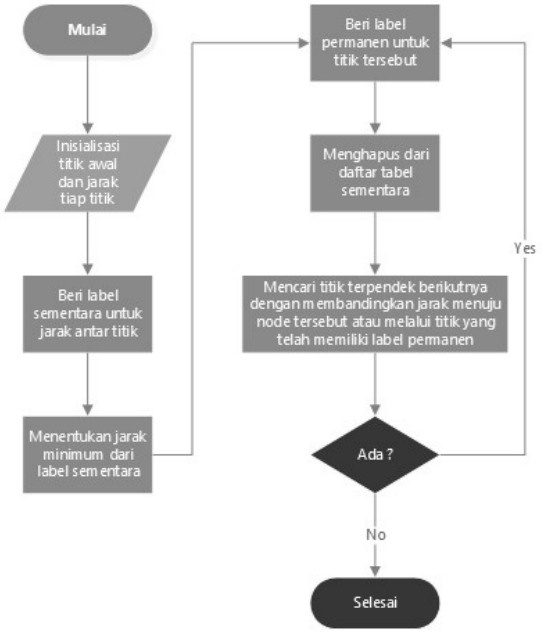
1. Data ke-i dibandingkan dengan data ke-(i+1), apabila tidak sesuai maka data akan ditukar untuk menempati urutan yang benar, yaitu data ke-i = data ke-(i+1) dan data ke-(i+1) = data ke-i.
2. Setelah menempati urutan yang benar, maka selanjutnya akan diurutkan dengan i-1 untuk memperoleh nilai terkecil, dan dibandingkan dengan data berikutnya: data ke-(i+1) dengan

data-data selanjutnya (i++), dan data tersebut akan ditukar jika tidak sesuai urutannya. Proses perbandingan data ini akan diulang kembali dan dilanjutkan sampai data terakhir.

3. Iterasi dilakukan sampai beberapa kali, sampai diperoleh suatu kondisi tidak ada lagi pertukaran data dalam satu iterasi.

2.2 Penentuan Waktu Tempuh

Setelah penentuan jarak ke halte terdekat maka penyelesaian dari masalah penelitian ini akan masuk pada tahap penentuan waktu tempuh ke jarak tersebut. Selanjutnya akan digunakan rumus perhitungan waktu menentukan waktu tempuh dari posisi pengguna ke halte terdekat.



Gambar 1. Flowchart mencari jarak halte terpendek dengan Algoritme Dijkstra

Untuk menentukan waktu tempuh menuju halte Transjakarta terdekat, digunakan kecepatan rata-rata pejalan kaki adalah 1.5 m/s (Mashuri, 2011). Sehingga waktu tempuh dapat dicari menggunakan rumus persamaan (2).

$$t = s / v \quad (2)$$

dimana:
t = waktu tempuh (s)
s = jarak (m)
v = kecepatan pejalan kaki (1.5 m/s)

Dengan begitu rumus ini dapat menghasilkan waktu dalam satuan detik (*seconds*). Agar lebih mudah dipahami maka hasil perhitungan di atas dikonversi ke dalam bentuk jam dan menit sebagai berikut.
Jam = jarak/3600
Menit = ((jarak % 3600)/60)

3. HASIL DAN PENGUJIAN

3.1 Hasil Rancangan Aplikasi

Langkah-langkah penggunaan aplikasi adalah sebagai berikut.

1. Saat GPS *smartphone* pengguna diaktifkan, aplikasi akan menerima titik koordinat pengguna berupa titik latitude dan longitude.
2. Setelah itu maka proses selanjutnya adalah menyimpan titik koordinat GPS sebagai titik awal.
3. Langkah berikutnya adalah menetapkan titik-titik lokasi tujuan dengan memasukkan titik koordinat halte-halte TransJakarta secara manual sesuai tempat yang diinginkan.
4. Menyimpan titik-titik koordinat halte TransJakarta sebagai titik tujuan.
5. Tahap berikutnya akan dibantu oleh pembendaharaan data dari *Google API Client Libraries* dimana pembentukan *edge* dan *graph* terbentuk otomatis dalam *source code Google API* versi 2.
6. Penggunaan Google API akan membentuk graph secara otomatis dari titik keberangkatan menuju titik tujuan yang terdiri dari beberapa edge.
7. Kemudian edge tersebut dibandingkan menggunakan algoritme Djikstra untuk diambil jalur dengan bobot terkecil. Hasil perbandingan ini akan disimpan dalam label sementara dan diulangi prosesnya sampai terbentuk jalur terpendek dan dimasukkan ke dalam tabel.
8. Berbeda dengan Algoritme Dijkstra yang digunakan untuk melukis jalur sebenarnya pada peta maka Haversine Formula digunakan untuk menampilkan hasil perhitungan jarak terdekat pada tampilan *interface* dengan rumus pendekatan tanpa mempertimbangkan jalur sebenarnya.
9. Bobot yang telah didapat dari hasil perhitungan Haversine Formula akan diurutkan dari yang terkecil sampai terbesar menggunakan algoritme Bubble Sort. Metode ini dilakukan dengan membandingkan bobot pertama dengan bobot selanjutnya dan begitu seterusnya.
10. Proses selanjutnya adalah menampilkan urutan halte TransJakarta terdekat dari posisi penumpang

3.2 Komponen Pengembangan dan Pengimplementasian Aplikasi

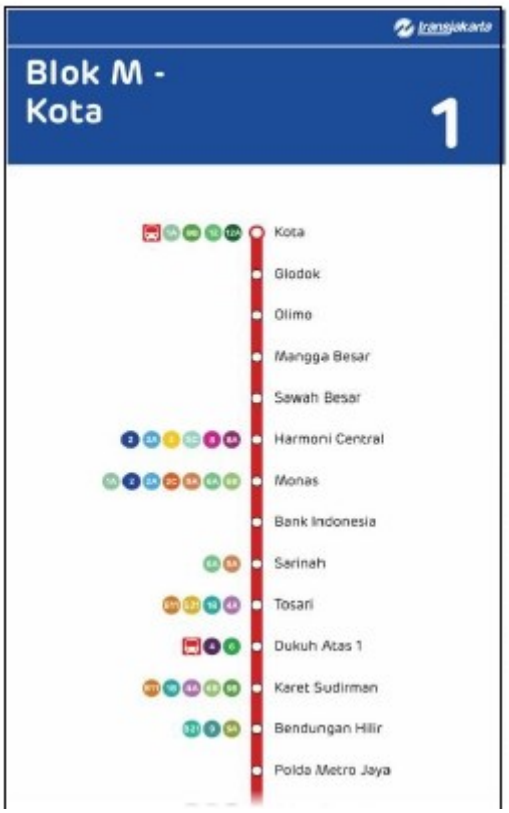
Processor : AMD Dual Core E1-2500
Memori RAM : 2 GB
Hard disk : 500GB 5400RPM
Peripheral : *Mouse*, *Keyboard*, *Monitor*, jaringan yang terhubung
Sistem Operasi : Windows 8.1 64-OS
Bahasa Pemrograman : Java dan XML
Tools Pengembangan : Eclipse 3.8, Notepad
: *Java SE Development Kit 8 Update 25* (8u25)
: Android *SDK*, *ADT bundle*
Browser : Google Chrome
Smartphone : Android Xiomi 3X
Sistem operasi : Android 6.0.1 (Marsmallow)
Chipset : Qualcomm MSM8937 Snapdragon 430
CPU : Octa-core 1.4 GHz Cortex-A53
GP : Adren

3.3 Tampilan Aplikasi

3.3.1 Tampilan Menu Utama

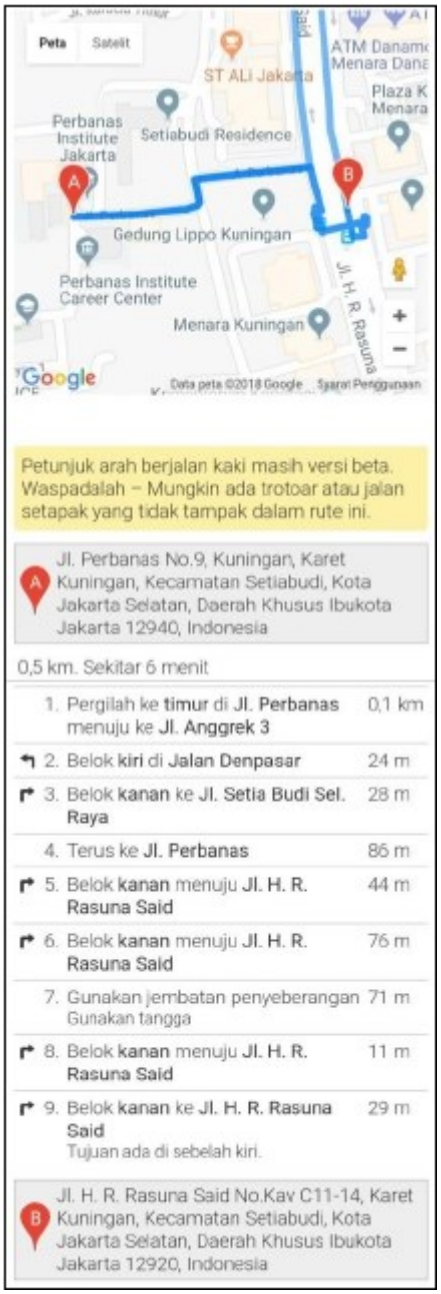
Aplikasi ini memiliki empat menu utama yang terdiri dari menu *Map, Corridor list, Search* dan *About.*

3.3.2 Tampilan Menu Corridor List



Gambar 2. Tampilan Halte Koridor Transjakarta

Pada menu ini pengguna dapat mengetahui informasi mengenai rute apa saja yang akan dilalui oleh bus, tempat transit, pemberhentian, maupun tujuan akhir melalui menu *Corridor List*.



Gambar 3. Tampilan Pilih Metode Menuju Halte Transjakarta

Apabila menu Map dibuka maka akan ditampilkan daftar halte TransJakarta yang telah diurutkan berdasarkan jarak dari yang terdekat sampai terjauh dari posisi pengguna saat itu. Selain itu juga ditampilkan estimasi waktu yang disesuaikan dengan kecepatan pejalan kaki menuju halte tujuan. Tampilan ini akan menunjukkan cara Algoritme Dijkstra mengarahkan pengguna menuju halte terdekat. Pengguna tidak perlu kuatir apabila Wi-Fi atau data seluler tidak berfungsi karena metode ini akan tetap menunjukkan rute yang perlu ditempuh

dengan deskripsi perjalanan yang dimilikinya. Aplikasi ini juga dilengkapi dengan pemilihan jalur alternatif, dan dimaksudkan agar pengguna dapat memilih jalur yang sesuai dengan keinginannya atau situasi saat itu.

3.4 Pengujian

3.4.1 Pengujian Aplikasi

Pengujian 1 dilakukan dengan cara mencoba satu persatu menu yang ada pada aplikasi yang hasilnya ditampilkan pada Tabel 1.

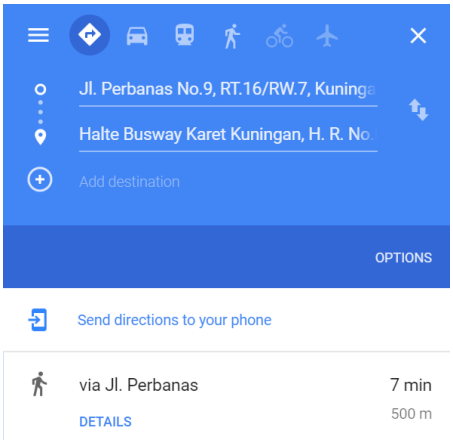
Tabel 1. Pengujian Aplikasi

Pengujian 1 - Menampilkan menu Stop Near Me	
Deskripsi	Berisi daftar rekomendasi halte Transjakarta terdekat
Hasil	- dapat mengeluarkan daftar rekomendasi halte Transjakart terdekat sampai terjauh - dapat menampilkan estimasi jarak yang dibutuhkan - dapat menampilkan waktu tempuh menuju halte Transjakarta terdekat - dapat menampilkan deskripsi rute menuju halte Transjakarta - dapat terhubung dengan Google Maps
Pengujian 2 - Menampilkan menu Map	
Deskripsi	Berisi peta yang terhubung langsung ke Google Maps
Hasil	- dapat menampilkan posisi GPS pengguna - dapat menampilkan titik koordinat halte Transjakarta sekitar pengguna - dapat mengarahkan pejalan kaki menuju halte Transjakarta
Pengujian 3 - Menampilkan menu Corridor List	
Deskripsi	Berisi daftar koridor halte TransJakarta
Hasil	- dapat menampilkan daftar koridor Transjakarta - dapat menampilkan informasi mengenai koridor halte TransJakarta

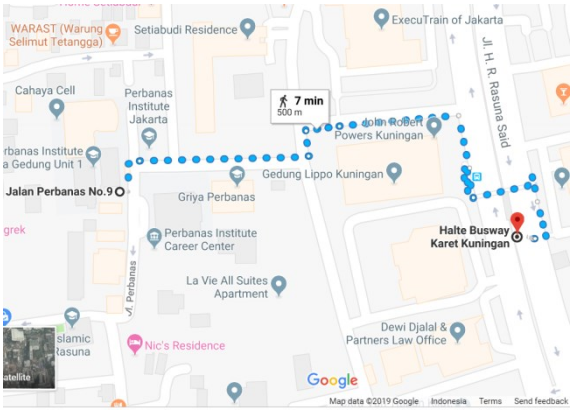
3.4.2 Pengujian perbandingan Aplikasi Djikstra dengan Google Map

Pengujian selanjutnya hanya mengambil satu menu rute yang diperoleh dari aplikasi (Gambar 3) dengan hasil yang diperoleh menggunakan Google Map. Pengujian dengan menggunakan Google Map akan menggunakan titik awal yang muncul pada aplikasi dan titik akhir sesuai urutan halte terdekat yang muncul.

Dari hasil pencarian rute menggunakan aplikasi dan yang diperoleh pada Google Map untuk dua titik yang sama hasilnya dapat dilihat pada Tabel 2.



Gambar 4. Hasil yang muncul pada Google Map terlihat pada



Gambar 5. Rute yang diperoleh dari Google Map

Tabel 2 Perbandingan hasil

Indikator	Aplikasi	Google Map
Jarak	500 meter	500 meter
Waktu tempuh	6 menit	7 menit
Rute	Lebih detail	Sesuai dengan standar Google Map

Analisis hasil perbandingan antara aplikasi Djikstra dengan Google Map adalah:

1. Jarak antara dua titik pada Aplikasi Djikstra dan Google Map menunjukkan angka yang sama.
2. Ada selisih waktu 1 menit untuk waktu tempuh, hal ini dimungkinkan terjadi karena acuan yang digunakan untuk menghitung kecepatan berbeda.
3. Rute pada Aplikasi Djikstra lebih detil dibandingkan dengan Google Map.

4. KESIMPULAN

Aplikasi penentuan rute dan waktu tempuh ke halte Transjakarta dengan Algoritme Dijkstra berbasis mobile dapat memberikan gambaran perjalanan menuju halte terdekat. Aplikasi juga dapat menampilkan daftar rekomendasi halte Transjakarta

dari sisi pengguna sehingga informasi yang dibutuhkan tersedia dengan tepat. Aplikasi ini juga dapat memberikan rute alternatif lainnya pada pengguna. Kelebihan aplikasi ini adalah *Wi-Fi* atau data seluler mati karena metode ini akan tetap menunjukkan rute yang perlu ditempuh dengan deskripsi perjalanan yang dimilikinya.

Pengembangan aplikasi selanjutnya diharapkan dapat menentukan halte Transjakarta terdekat berdasarkan tujuan yang ingin dituju seperti nama daerah, jalan, gedung, maupun tempat umum lainnya. Aplikasi selanjutnya juga diharapkan dapat menambahkan fitur yang dapat membantu pengguna memilih halte Transjakarta terdekat yang tepat untuk melakukan transit sesuai tujuan. Aplikasi juga harus selalu dimutakhirkan basisdata koridor dan haltenya, sehingga daftar halte Transjakarta yang dapat ditempuh pengguna bertambah untuk memperluas jangkauan peta.

DAFTAR PUSTAKA

BUDIHARTANTI, C., & PANDIANGAN, R. .2016. Rancang Bangun Aplikasi Android Pencarian Rumah Sakit Di Jakarta Menggunakan Algoritma Dijkstra. *Jurnal PROSISKO*, *3*(2), 1–8.

CHEN, Z., SHEN, H. T., ZHOU, X., & YU, J. X. 2009. Monitoring path nearest neighbor in road networks. *Proceedings of the 35th SIGMOD International Conference on Management of Data - SIGMOD '09*, 591. http://doi.org/10.1145/1559845.1559907

CHOPDE, N. R., & NICHAT, M. K. 2013. Landmark based shortest path detection by using Dijkestra Algorithm and Haversine Formula. *International Journal of Innovative Research in Computer Engineering*, *1*(2), 298–302. http://doi.org/10.1.1.300.5943

DWI, SAPUTRA, R., & ARDANA. 2016. Penerapan Algoritma Dijkstra pada Aplikasi Pencarian Rute Bus Trans Semarang. *Seminar Nasional Ilmu Komputer*, 299–306.

GALÁN-GARCÍA, J. L., AGUILERA-VENEGAS, G., GALÁN-GARCÍA, M., & RODRÍGUEZ-CIELOS, P. 2015. A new Probabilistic Extension of Dijkstra's Algorithm to simulate more realistic traffic flow in a smart city. *Applied Mathematics and Computation*, *267*, 780–789. http://doi.org/10.1016/j.amc.2014.11.076

GUNAWAN, K. 2015. Implementation of Location Base Service on Tourism Places in West Nusa Tenggara by using Smartphone. *International Journal of Advanced Computer Science and Applications*, *6*(8), 160–166. Retrieved from http://bep.ejurnal.net/index.php/int/article/view/14

GUSMÃO, A., PRAMONO, S. H., & SUNARYO. 2013. Sistem Informasi Geografis Pariwisata Berbasis Web Dan Pencarian Jalur Terpendek Dengan Algoritma Dijkstra. *Jurnal Electrics, Electronics, Communications, Controls, Informatics, Systems*, *7*(2), 125–130.

KUMALASARI, D. 2017. Analisis Perbandingan Kompleksitas Algoritma Bubble Sort, Cocktail Sort dan Comb Sort Dengan Bahasa Pemrograman C ++. *Journal Speed*, *9*(2), 1–7.

MASHURI, I. M. 2011. Studi Karakteristik Pejalan Kaki dan Pemilihan Jenis Fasilitas Penyeberangan Pejalan Kaki di Kota Palu (Studi Kasus: Jl. Emmi Saelan Depan Mal Tatura Kota Palu). *Journal of Transportation Management and Engineering*, *1*(2), 67–79.

PRASETYO, D., & HASTUTI, K. 2015. Penerapan Haversine Formula pada Aplikasi Pencarian Lokasi dan Informasi Gereja Kristen di Semarang Berbasis Mobile. *Skripsi Teknik Informatika Universitas Dian Nuswantoro*.

RAHAYUNINGSIH, P. A. 2016. Analisis Perbandingan Kompleksitas Algoritma Pengurutan Nilai (Sorting). *Jurnal Evolusi*, *4*, 2016.

ROSYIDI, L., PRADITYO, H. P., GUNAWAN, D., & SARI, R. F. 2014. Timebase dynamic weight for Dijkstra Algorithm implementation in route planning software. *Proceedings of 2014 International Conference on Intelligent Green Building and Smart Grid, IGBSG 2014*, (July 2017). http://doi.org/10.1109/IGBSG.2014.6835261

SINGAL DCSA, P., & CHHILLAR, R. R. S. 2014. Dijkstra Shortest Path Algorithm using Global Positioning System. *International Journal of Computer Applications*, *101*(6), 975–8887. http://doi.org/10.15270/50-4-387

SONITA, A., & NURTANEO, F. 2015. Analisis Perbandingan Algoritma Bubble Sort, Merge Sort, dan Quick Sort dalam Proses Pengurutan Kombinasi Angka dan Huruf. *Jurnal Pseudocode*, *II*(September), 75–80.

SURYO SAPUTRO, S. 2013. Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata Di Kota Manado Berbasis Mobile Web Dengan Algoritma Dijkstra. *Journal of Chemical Information and Modeling*, *53*(9), 1689–1699. http://doi.org/10.1017/CBO9781107415324.004

Halaman ini sengaja dikosongkan