

Aplikasi Perhitungan Dan Visualisasi Jarak Terpendek Berdasarkan *Data Coordinate* Dengan Algoritma *Dijkstra* Dalam Kasus Pengantaran Barang Di Kawasan Jabodetabek

Elsa Kusuma¹, Jefri², Halim Agung³
Program Studi Teknik Informatika^{[1],[2],[3]}
Universitas Bunda Mulia
Jakarta, Indonesia

Abstrak— Penelitian diangkat berdasarkan rute-rute yang ditempuh para pengantar barang. Karena kondisi jalan yang macet dan memakan waktu lama, mereka seringkali kebingungan untuk menentukan rute terbaik. Berdasarkan variasi rute yang ditempuh dan jarak antara titik awal dan tujuan, kita dapat menghitung dan menentukan, jalur mana yang memiliki jarak terpendek dan terefektif. Dalam penelitian ini, algoritma *Dijkstra* diterapkan untuk menghitung jarak terpendek dari suatu titik ke titik lainnya dalam area Jabodetabek, berdasarkan banyaknya titik koordinat yang terdapat dalam *database*. Metode *Iterative-Enhancement* diterapkan dalam mengembangkan sistem ini. Pengumpulan data dilakukan melalui observasi pola rute yang sering dilalui dan studi literatur, disertai penelitian terdahulu tentang implementasi algoritma *Dijkstra*. Sistem dirancang dalam beberapa tahapan, yaitu analisis kebutuhan sistem dan *user*, perancangan *flowchart*, perancangan *DFD*, perancangan *database*, hingga perancangan tampilan sistem. Hasil penelitian akan diwujudkan dalam bentuk sistem perangkat lunak berbasis *web* dan mencantumkan peta sederhana sebagai sarana untuk membantu para pengantar barang menentukan rute pengantaran yang lebih sesuai. Hasil penelitian diperoleh dari pengujian dengan dua kondisi (jumlah data) yang berbeda, yakni 2.056 dan 5.186 titik koordinat. Pengujian menunjukkan perbedaan jarak terpendek yang signifikan untuk masing-masing kondisi, di mana hasil perhitungan dengan kondisi jumlah data lebih sedikit tidak lebih efektif dibandingkan kondisi jumlah data lebih banyak. Dari pembahasan dan pengujian, disimpulkan bahwa algoritma *Dijkstra* sangat membantu dalam menentukan rute terpendek yang optimal. Didapati pula bahwa jumlah data sangat berpengaruh terhadap hasil perhitungan. Semakin banyak jumlah titik koordinat yang ada dalam *database*, semakin optimal hasil perhitungan yang akan diperoleh. Sistem juga menawarkan kemudahan bagi setiap pengguna melalui tampilan peta yang tertera sebagai visualisasi rute.

Kata Kunci— *Jabodetabek, Dijkstra, Iterative-Enhancement, Titik koordinat, Latitude, Longitude, Euclidean Distance, Jarak terpendek*

I. PENDAHULUAN

Dewasa ini, mobilitas di jalan raya kerap kali menjadi kendala di kota-kota besar, khususnya Jakarta. Kemacetan di jalan utama yang terlalu padat menambah parah kondisi tersebut. Banyak dari pengendara jalan, akhirnya memilih rute alternatif untuk mencapai destinasinya. Hal ini membuat rute yang ditempuh menjadi beraneka ragam dan perlu untuk diteliti, mana yang paling efektif.

Kasus yang penulis angkat kali ini didasari pada rute-rute yang ditempuh para pengantar barang. Dengan tujuan untuk tiba di destinasi pengiriman secepat mungkin, banyak dari mereka yang begitu mengenal kawasan lalu lintas. Pengetahuan mereka tersebut dipergunakan untuk mempercepat layanan mereka mengirim barang.

Berdasarkan variasi rute yang ditempuh dan pola cakupan pengantaran yang identik, kita dapat menghitung dan menentukan, jalur mana yang memiliki jarak terpendek dan lebih efektif. Nantinya, hasil tersebut dapat direkomendasikan kepada para pengantar barang. Kecepatan pengiriman pun akan meningkat yang kemudian akan berbuah pada banyak hal positif lainnya.

Oleh karena masalah tersebut, penulis ingin membuat penelitian yang berjudul Penentuan Jarak Terpendek Berdasarkan Data Coordinate Menggunakan Algoritma *Dijkstra* Dalam Kasus Pengantaran Barang Se-Jabodetabek. Lintasan terpendek dapat diartikan sebagai bobot minimal dari kombinasi rute. Grafis lintasan tersebut yang akan dijadikan indikator hasil perhitungan jarak terpendek.

II. LANDASAN TEORI

A. Teori Transportasi

Pengertian transportasi berasal dari kata Latin, yaitu *transportare*, di mana *trans* diartikan seberang atau sebelah

lain, dan *portare* berarti mengangkut atau membawa. Jadi secara harafiah, transportasi berarti mengangkut atau membawa (sesuatu) ke sebelah lain, dari suatu tempat ke tempat lain. Ini berarti transportasi merupakan suatu jasa yang diberikan, guna menolong orang atau barang dalam hal pemindahan posisi. Dapat ditegaskan pula bahwa transportasi adalah jasa yang dipergunakan sebagai alat untuk memperoleh keuntungan-keuntungan ekonomis dalam berbagai kegiatan usaha dan hubungan kemasyarakatan^[5].

B. Latitude dan Longitude

Latitude adalah garis yang melintang di antara kutub utara dan kutub selatan, yang menghubungkan antara sisi timur dan barat bagian bumi. Garis ini memiliki posisi membentangi bumi, sama halnya seperti garis *equator* (khatulistiwa), tetapi dengan kondisi nilai tertentu. Garis lintang inilah yang dijadikan ukuran dalam mengukur sisi utara-selatan koordinat suatu titik di belahan bumi. Sedangkan, *longitude* adalah garis membujur yang menghubungkan antara sisi utara dan sisi selatan bumi (kutub). Garis bujur ini digunakan untuk mengukur sisi barat-timur koordinat suatu titik di belahan bumi. Sama seperti *equator* pada *latitude* yang berada di tengah dan memiliki nilai 0 (nol) derajat, pada *longitude*, garis tengah yang bernilai 0 (nol) derajat disebut garis *prime meridian* (garis bujur). Sedangkan, garis yang berada paling kiri memiliki nilai -90 derajat, dan yang paling kanan memiliki nilai 90 derajat^[12].

C. Euclidean Distance

Euclidean Distance adalah metrik yang digunakan untuk menghitung kesamaan 2 vektor. Fungsi *Euclidean* menghitung akar dari kuadrat perbedaan 2 vektor. *Euclidean space* diperkenalkan oleh Euclid, seorang matematikawan dari Yunani sekitar tahun 300 B.C.E. untuk mempelajari hubungan antara sudut dan jarak. *Euclidean* ini biasanya diterapkan pada 1, 2, dan 3 dimensi^[12]. Adapun rumus dari perhitungan *Euclidean Distance* dalam ranah 2 dimensi sebagai berikut:

$$\sqrt{(Latitude1 - Latitude2)^2 + (Longitude1 - Longitude2)^2}$$

Rumus 2.1 Perhitungan *Euclidean Distance*^[12]

D. Algoritma

Algoritma merupakan langkah-langkah logis penyelesaian masalah yang disusun secara sistematis terhadap permasalahan yang akan diselesaikan. Dalam bidang komputer, algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemrograman, terutama dalam proses komputasi numerik. Tanpa algoritma yang dirancang dengan baik maka proses pemrograman akan menjadi salah, rusak atau lambat dan tidak efisien^[6].

- Algoritma Pencarian Rute Terpendek
Secara umum, pencarian jalur terpendek dibagi menjadi dua metode, yaitu metode

konvensional dan metode heuristik. Metode konvensional cenderung mudah dipahami dari pada metode heuristik. Tetapi bila dibandingkan, hasil yang diperoleh dari metode heuristik lebih variatif dan waktu yang diperlukan lebih singkat.

- Algoritma Dijkstra

Pencarian rute terpendek termasuk ke dalam materi teori graf. Algoritma yang sangat terkenal untuk menyelesaikan persoalan ini adalah algoritma *Dijkstra*. Algoritma ini ditemukan oleh seorang ilmuwan komputer berkebangsaan Belanda yang bernama Edsger Dijkstra. Algoritma *Dijkstra* dianggap cocok karena mudah digunakan oleh *user* dalam penggunaannya hanya dengan menentukan titik awal dan titik tujuan^[14].

Dijkstra adalah algoritma yang digunakan untuk mencari lintasan terpendek pada sebuah graf berarah. Contoh penerapan algoritma ini adalah lintasan terpendek yang menghubungkan antara dua kota berlainan tertentu. Kasus ini sering disebut *Single-source Single Destination Shortest Path Problems*.

Cara kerja algoritma *Dijkstra* memakai strategi *greedy*, di mana pada setiap langkah dipilih sisi dengan bobot terkecil yang menghubungkan sebuah simpul yang sudah terpilih dengan simpul lain yang belum terpilih. Algoritma *Dijkstra* membutuhkan parameter berupa tempat asal dan tempat tujuan^[4].

Parameter tersebut berisi rute berbentuk sisi (*vertex*) atau *vertices* dalam bentuk jamak untuk diperbandingkan. Setiap sisi dari rute ini adalah pasang *vertices(u,v)* yang melambangkan hubungan dari *vertex* u ke *vertex* v. Himpunan semua tepi kita sebut sebagai *E*. Bobot (*weights*) dari semua sisi dihitung dengan fungsi :

$$w : E \rightarrow [0, \infty]$$

Rumus 2.2 Perhitungan Bobot pada Algoritma *Dijkstra*^[13]

Jadi, $w(u,v)$ adalah jarak tak-negatif dari *vertex* u ke *vertex* v. Ongkos (*cost*) dari sebuah sisi dapat dianggap sebagai jarak antara dua *vertices*, yaitu jumlah jarak semua sisi dalam jalur tersebut^[9]. Untuk sepasang *vertex* s dan t dalam V, algoritma ini menghitung jarak terpendek dari s ke t. Adapun *pseudocode* penerapannya dapat divisualisasikan sebagai berikut :

```

Procedure dijkstra (w, a, z, L)
  L(a) := 0
  S := {}
  for semua vertex x≠a do
    L(x) := ∞
  T := himpunan semua vertex
  while z(T) do
    begin
    pilih v(T) dengan minimum L(v)
    T := T-{v}
    S := S union {v}
    for setiap x(T) di samping v do
      L(x) := min{L(x), L(v)+w(v,x)}
    end
  end dijkstra
  
```

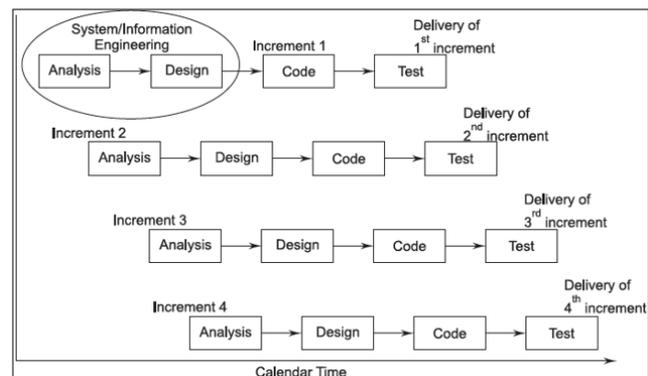
Gambar 2.1 Pseudocode Algoritma Dijkstra^[7]

Algoritma *Dijkstra* diterapkan pada penelitian berjudul *Vehicle Routing Problems for City Logistics*[2] dengan hasil penelitian bahwa optimasi rute kendaraan berperan sebagai kunci dalam perkembangan mobilisasi produk urban. Penelitian berjudul *Aplikasi pgRouting Untuk Penentuan Jalur Optimum Pada Pembuatan Rute Pemadam Kebakaran (Studi Kasus : Kota Semarang)* [3] menghasilkan tabel spasial/layer baru melalui beberapa perintah *SQL Query* yang tersimpan pada basis data *PostgreSQL*. Penelitian berjudul *Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra Pada Sistem Informasi Geografis Pemetaan Stasiun Pengisian Bahan Bakar Umum*[4] menghasilkan informasi dan membantu pengguna dalam melakukan pencarian lokasi SPBU berdasarkan nama dan rute terpendek antar lokasi SPBU dalam bentuk peta jalan kota Padang. Penelitian berjudul *An Application of Dijkstra's Algorithm to Shortest Route Problem*[7] menghasilkan Rute terpendek antar *node* dalam jaringan peta dapat ditentukan dari destinasi yang diinginkan dan melalui proses *back-tracking* melewati *node-node* yang ada. Penelitian berjudul *Railway Route Optimization System Using Dijkstra Method*[8] menghasilkan kesimpulan rute terpendek yang diperoleh akan lebih optimal berdasarkan jarak di antara dua stasiun, bukan waktu yang ditempuh dari stasiun awal ke stasiun akhir. Penelitian berjudul *Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra Pada SIG Berbasis Web Untuk Distribusi Minuman (Studi Kasus PT. Coca Cola Kota Padang)* [9] menghasilkan informasi untuk PT. Coca Cola tentang jalur mana yang bisa ditempuh untuk mempercepat proses pendistribusian engan mengikuti jalur terpendek yang dihasilkan. Penelitian berjudul *Pencarian SPBU Terdekat Dan Penentuan Jarak Terpendek Menggunakan Algoritma Dijkstra (Studi Kasus di Kabupaten Jember)* [10] menghasilkan perhitungan jarak terpendek dalam pencarian SPBU

dipengaruhi oleh nilai kriteria, *cost*, dan *reverse cost*. Sedangkan, nilai waktu tempuh didapat dari perhitungan jarak dibagi kecepatan. Penelitian berjudul *Optimalisasi Algoritma Dijkstra Dalam Menghadapi Perbedaan Bobot Jalur Pada Waktu Yang Berbeda* [13] menghasilkan kesimpulan bahwa keterbatasan algoritma *Dijkstra* dapat diatasi dengan penerapan metode yang diusulkan dalam menghadapi perbedaan bobot jalur yang berbeda-beda.

E. Iterative-Enhancement Model

Software-development life-cycle adalah proses terbentuknya sebuah sistem informasi yang dimulai dari konseptual ke implementasi. *Software-development life-cycle* digunakan untuk membantu mengerti keseluruhan proses, membantu untuk merencanakan sumber daya yang digunakan, dan membantu untuk mengontrol jalannya proses. *Software-development life-cycle* terbagi menjadi lima sampai sembilan tahap, minimal harus memiliki lima tahap dan paling banyak adalah sembilan tahap^[6].



Gambar 2.2 Iterative-Enhancement Model^[6]

Iterative-enhancement model adalah model linier yang diterapkan secara berulang-ulang. Pada model ini, proses dipecah menjadi beberapa modul yang dikembangkan secara bertahap. Kebutuhan utama yang telah diketahui dapat diproses terlebih dahulu, kebutuhan termasuk perubahan lainnya dapat diproses setelahnya. Masing-masing proses dapat dilakukan berulang-ulang hingga mendapatkan hasil yang diinginkan^[6].

F. Data Flow Diagram (DFD)

Data Flow Diagram digunakan untuk menggambarkan suatu fungsi dari suatu sistem informasi. Beberapa simbol yang digunakan dalam *Data Flow Diagram* dapat dilihat pada Tabel 2.1.

Tabel 2.1 Simbol DFD^[14]

Simbol	Nama dan Arti
	Function Symbol, untuk melakukan pengolahan beberapa input data
	External Entity, untuk menggambarkan entitas diluar system yang berhubungan langsung dengan sistem.
	Data Flow Symbol, untuk menggambarkan aliran proses.
	Data Store Symbol, untuk menggambarkan penyimpanan data.
	Output Symbol, untuk menggambarkan akuisisi data.

Data Flow Diagram memiliki beberapa tahap dalam pembuatannya. Tahap awal adalah diagram konteks yang menggambarkan keseluruhan sistem secara umum. Selanjutnya adalah diagram level 0 untuk menggambarkan tahapan proses dari diagram konteks. Selanjutnya adalah diagram level 1 untuk menggambarkan tahapan proses dari diagram level 0 dan seterusnya^[14].

G. Entity Relationship Diagram (ERD)

ERD (Entity Relationship Diagram) adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam sistem secara abstrak. Dalam sebuah sistem informasi entitas adalah data yang mewakili seseorang, tempat, benda, atau kegiatan untuk disimpan atau diolah, sedangkan karakteristik khusus atau ciri-ciri khusus dalam suatu entitas disebut atribut. Sebuah database dapat memiliki lebih dari satu entitas dan hubungan antar entitas tersebut disebut ERD. [14]

ERD merupakan model yang menunjukkan relasi logis dan interaksi antar entitas. ERD digunakan untuk memodelkan keseluruhan sistem dan dasar untuk membuat struktur data fisik. Simbol-simbol yang digunakan dalam ERD dapat dilihat pada Tabel 2.2.

Tabel 2.2 Simbol ERD Crow's Foot^[14]

Simbol	Nama dan Representasi	Representasi UML
	One and Only One (1)	Hanya satu anggota entitas yang berrelasi dengan entitas lain
	One or Many (1..*)	Bisa terdapat satu atau lebih anggota entitas yang berrelasi dengan entitas lain
	Zero, or One, or Many (0..*)	Bisa terdapat satu, atau lebih, atau tidak sama sekali anggota entitas yang berrelasi dengan entitas lain
	Zero, or One (0..1)	Bisa terdapat satu atau tidak sama sekali anggota entitas yang berrelasi dengan entitas lain

H. Flowchart

Flowchart adalah sebuah teknik yang merepresentasikan alir kendali dalam program [6]. Beberapa simbol yang digunakan dalam flowchart dapat dilihat pada Tabel 2.3.

Tabel 2.3 Simbol Flowchart^[6]

Simbol	Nama dan Arti
	Terminal Symbol, untuk mendefinisikan awal dan akhir flowchart
	Input/ Output Symbol, untuk mengindikasikan pemosisian logis operasi input/ output.
	Processing Symbol, untuk mengindikasikan operasi aritmatika ataupun perpindahan data.
	Decision Symbol, untuk menggambarkan keputusan yang diambil.
	Flow Lines, adalah garis dengan anak panah yang digunakan untuk mengindikasikan aliran logika program dan menunjuk pernyataan selanjutnya yang akan dieksekusi.
	Connector Symbol, jika sebuah flowchart dipotong menjadi beberapa bagian, connector symbol digunakan untuk menghubungkan bagian-bagian tersebut.
	Hexagon, merupakan kotak persiapan yang berisi pernyataan aturan pengulangan.

III. ANALISIS DAN PERANCANGAN

A. Analisis Kebutuhan

Berikut ini adalah kebutuhan fungsional dan kebutuhan non-fungsional dari sistem penentuan jarak terpendek ini :

1) Kebutuhan fungsional

Kebutuhan fungsional yang ada dalam sistem ini adalah sebagai berikut:

- a) Sistem dapat menerima data masukan dari user berupa titik awal penerimaan barang dan titik destinasi pengiriman barang.
- b) Sistem dapat mengkonversi masukan tersebut menjadi format koordinat *latitude* dan *longitude*.
- c) Sistem dapat terkoneksi dengan database penyimpanan histori rute para pengantar barang.
- d) Sistem dapat mengeksekusi masukan dan mengkalkulasi jarak terpendek berdasarkan *data coordinate*.
- e) Sistem dapat memberikan keluaran berupa nilai jarak terpendek dari titik awal ke titik akhir dalam satuan kilometer.

2) Kebutuhan non fungsional

Kebutuhan non-fungsional yang ada dalam sistem penentuan jarak terpendek ini adalah sebagai berikut :

- a) Beberapa perangkat lunak yang dibutuhkan untuk pengembangan sistem penentuan jarak terpendek ini adalah Sistem Operasi, minimal *Windows XP* atau versi di atasnya, *web browser*, dan *MongoDB*.
- b) Untuk dapat menjalankan sistem penentuan jarak terpendek ini dengan baik, dibutuhkan spesifikasi minimal perangkat keras adalah *Processor Intel Pentium IV*, RAM 2GB, dan Ruang *harddisk* kosong 1GB.

- c) Adapun analisis kebutuhan *user* adalah *User* yang memiliki kemampuan untuk mengoperasikan komputer dan memahami sistem operasi *Windows*, Sistem dapat memberikan notifikasi kepada *user* saat terjadi kesalahan pemasukan data atau ketidaklengkapan data, dalam kasus lain, sistem dapat memberikan notifikasi kepada *user* pula jika basis data belum terintegrasi dengan sistem yang menyebabkan kegagalan sistem.

B. Metodologi Pengembangan Sistem

Tahapan yang penulis lakukan dengan metodologi pengembangan sistem *iterative-enhancement*, yaitu:

- **Analysis**

Pada tahap ini, penulis melakukan studi kepustakaan yang bersumber dari buku, jurnal, skripsi dan internet. Lalu memutuskan untuk membuat sistem penentuan jarak terpendek berdasarkan *data coordinate* dengan menerapkan fungsi *Euclidean Distance* dan mengimplementasikan algoritma *Dijkstra*. Lalu penulis juga melakukan analisis dan perencanaan terhadap sistem yang dikembangkan. Kegiatan analisa dan perencanaan yang dilakukan antara lain, analisis kebutuhan fungsional sistem, analisis kebutuhan non-fungsional sistem, dan analisis kebutuhan *user*.

- **Design**

Pada tahap ini, dilakukan desain terhadap sistem penentuan jarak terpendek yang akan dikembangkan, antara lain desain alur kerja sistem dengan menggunakan *flowchart* dan desain fungsionalitas sistem dengan menggunakan *Data Flow Diagram*.

- **Code**

Pada tahap ini, penulis membangun sistem yang telah dirancang dengan menggunakan bahasa pemrograman *Javascript*.

- **Testing**

Pada tahap ini dilakukan uji coba sistem yang telah dibuat untuk melihat masih ada kesalahan atau tidak. Penulis melakukan pengecekan proses perhitungan jarak dan penentuan rute dari *data coordinate* berupa graf yang dapat dilihat oleh mata manusia. Penulis juga membandingkan optimasi *output* berupa jarak terpendek yang dihasilkan.

- **Delivery of Increment**

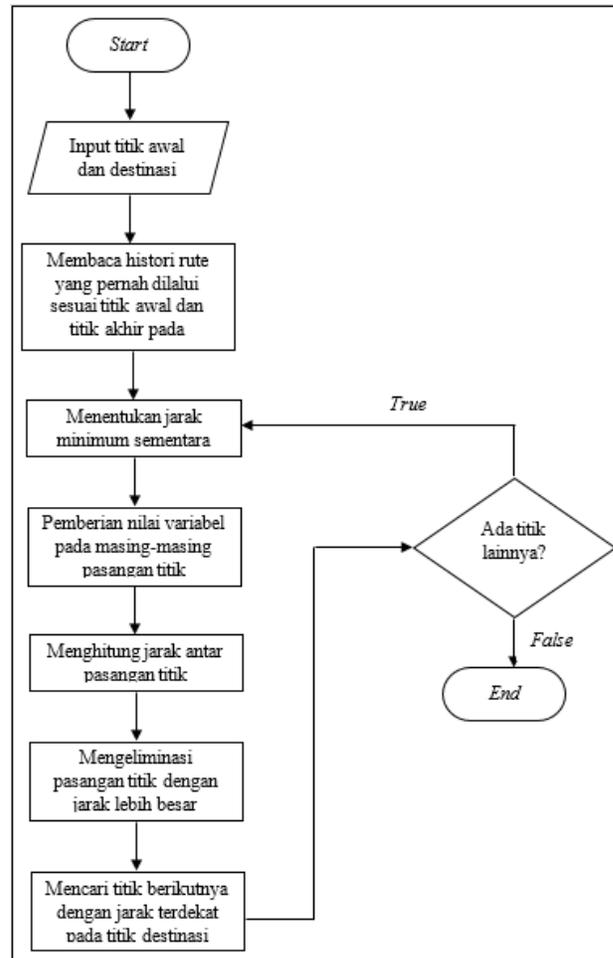
Pada tahap ini penulis menerima kritik dan saran dari para dosen mengenai mengenai sistem yang dibuat. Jika sistem disetujui, maka sistem akan dirilis. Tetapi, jika masih ada bagian dari sistem yang belum disetujui, maka penulis akan mengulang kembali proses

pengembangan aplikasi ke *increment* yang selanjutnya.

C. Perancangan Proses

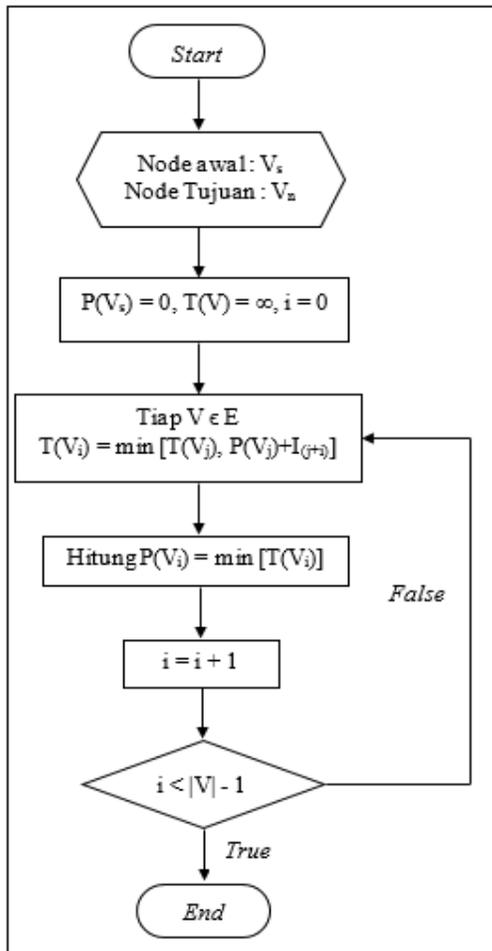
1) **Alur Penentuan Jarak Terpendek**

Proses perhitungan dan penentuan jarak dimulai dari penerimaan masukan berupa titik *pick-up* dan *delivery* dari *user*. Kemudian, sistem lanjut ke proses pencarian titik terdekat dan perhitungan jarak antar titik, hingga sampai ke titik destinasi (*delivery*). Alur proses digambarkan dalam bentuk *flowchart* pada gambar 3.1, sebagai berikut :



Gambar 3.1 *Flowchart* Proses Penentuan Jarak Terpendek

Sedangkan, proses perhitungan dengan algoritma *Dijkstra* digambarkan pada gambar 3.2, sebagai berikut :



Gambar 3.2 Flowchart Proses Perhitungan Jarak Terpendek

Secara deskriptif, algoritma ini terdiri dari beberapa langkah. Langkah-langkah algoritma *Dijkstra* dapat dipaparkan sebagai berikut:

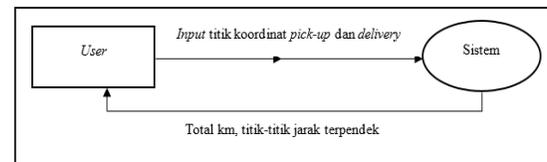
- Tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, *Dijkstra* akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap.
- Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi).
- Set semua node yang belum dilalui dan set node awal sebagai "Node keberangkatan".
- Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah dihitung nilainya) hapus data lama, simpan ulang data jarak dengan jarak yang baru.

- Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai "Node dilewati". Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
- Set "Node belum dilewati" dengan jarak terkecil (dari node keberangkatan) sebagai "Node Keberangkatan" selanjutnya dan ulangi langkah 5 hingga node tujuan tercapai.

3) DFD

a) Diagram Konteks

Diagram konteks pada sistem penentuan jarak terpendek yang dibuat dapat dilihat pada gambar 3.3.

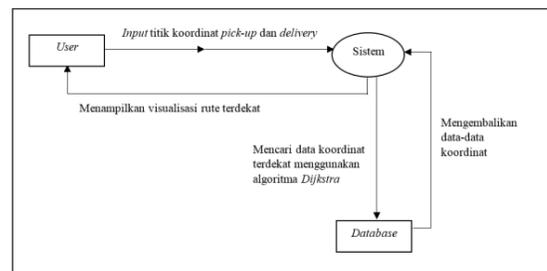


Gambar 3.3 Diagram Konteks Sistem Penentuan Jarak Terpendek

Pada Gambar 3.3, ditunjukkan bahwa terdapat 1 entitas di luar sistem, yaitu *user*. *User* dapat memberi masukan kepada sistem berupa titik koordinat *pick-up* (awal) dan *delivery* (destinasi). Sistem dapat menghasilkan keluaran berupa total kilometer jarak terpendek dan titik-titik terkait yang menunjukkan rutenya.

b) Diagram Level 0

Diagram *Level 0* pada sistem penentuan jarak terpendek yang dibuat dapat dilihat pada Gambar 3.4.



Gambar 3.4 Diagram Level 0 Sistem Penentuan Jarak Terpendek

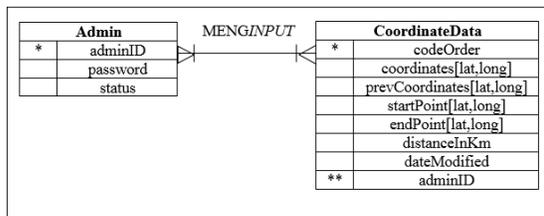
Pada Gambar 3.4, ditunjukkan bahwa terdapat 2 entitas di luar sistem, yaitu *user*

dan *database*. Sebelum proses perhitungan jarak dan pencarian titik, *user* memberi masukan kepada sistem berupa titik koordinat *pick-up* (awal) dan *delivery* (destinasi). Kemudian, sistem yang telah terkoneksi dengan *database*, mencari data koordinat terdekat menggunakan algoritma *Dijkstra*. Hasilnya, *database* mengembalikan data-data koordinat terdekat disertai posisi titiknya. Setelah mencapai titik *delivery*, sistem menampilkan visualisasi jarak terdekat dalam bentuk rangkaian titik kepada *user*.

D. Perancangan Basis Data

1) ERD

ERD pada sistem penentuan jarak terpendek ini terdiri dari satu entitas. Admin sebagai pembuat dan perubah data koordinat, serta *CoordinateData* yang memuat semua elemen yang berkaitan dengan perhitungan koordinat. Adapun ERD dapat dilihat pada Gambar 3.5, sebagai berikut :



Gambar 3.5 ERD Sistem Penentuan Jarak Terpendek

2) Kamus Data

Kamus data berisi daftar tabel, atribut, tipe, dan keterangan dari atribut yang akan menjadi indikator perhitungan jarak terpendek pada sistem ini. Adapun kamus data pada sistem ini dapat dilihat pada Tabel 3.1, sebagai berikut :

Tabel 3.1 Kamus Data Tabel *DataCoordinate*

Entitas	Elemen	Tipe Data	Keterangan
DataCoordinate (memuat data koordinat dan node)	codeOrder	string	Primary Key
	coordinate	[number]	Koordinat yang akan dihitung jaraknya
	prevCoordinate	[number]	Koordinat pembanding untuk menghitung jarak
	startPoint	[number]	Titik <i>pick-up</i> (awal) pengantaran barang
	endPoint	[number]	Titik <i>delivery</i> (destinasi) pengantaran barang
	distanceInKm	number	Jarak yang tercantum (satuan kilometer)

Tabel 3.2 Kamus Data Tabel Admin

Entitas	Elemen	Tipe Data	Keterangan
Admin (memuat informasi pengguna sistem)	adminID	string	Primary Key
	password	string	Sarana otorisasi ID
	status	string	Terdiri dari dua status, antara lain : 1. User, sebagai <i>end-user</i> 2. InputAdmin, sebagai penginput data ke <i>database</i>

IV. HASIL PENELITIAN

A. Halaman Sistem

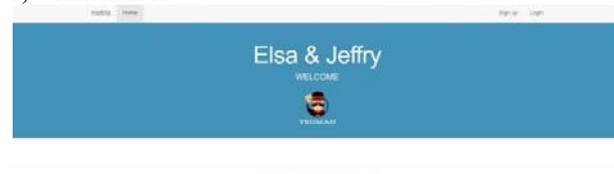
1) Halaman Login



Gambar 4.1 Halaman Login

Sesuai hasil perancangan, tertera area *input email* (*adminID*) dan *password* untuk masuk ke sistem. Tersedia pula opsi *sign up* untuk mendaftarkan ID baru.

2) Halaman Home



Gambar 4.2 Halaman Home

Halaman ini pun sesuai dengan perancangan yang menampilkan *adminID* dan menjadi halaman utama setelah proses *login* berhasil.

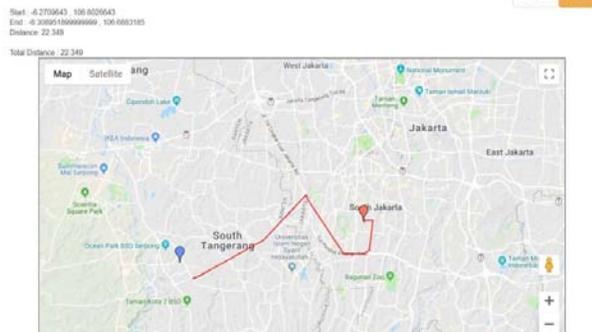
3) Halaman Input Titik Awal dan Akhir



Gambar 4.3 Halaman Input Titik Awal dan Akhir

Sesuai perancangan, halaman ini berperan bagi *user* untuk memberi masukan berupa titik awal dan tujuan. Setelah titik *diinput*, tombol *find* ditekan dan proses perhitungan jarak terpendek dimulai.

4) Halaman Hasil



Gambar 4.4 Halaman Hasil

Pada halaman ini berdasarkan perancangan, ditampilkan informasi titik koordinat awal, akhir, dan hasil perhitungan jarak terpendeknya. Peta sederhana menampilkan gambaran rute yang harus ditempuh. Simbol berwarna merah menunjukkan titik awal, sedangkan simbol berwarna biru menunjukkan titik tujuan.

B. Implementasi Algoritma Dijkstra

```
function dijkstraSolve(graph, s) {
  var solutions = {};
  solutions[s] = [];
  solutions[s].dist = 0;

  while (true) {
    var parent = null, nearest = null, dist = Infinity;

    //for each existing solution
    for (var n in solutions) {
      if (!solutions[n])
        continue
      var ndist = solutions[n].dist;
      var adj = graph[n];
      //for each of its adjacent nodes...
      for (var a in adj) {
        //without a solution already...
        if (solutions[a])
          continue;
        //choose nearest node with lowest *total* cost
        var d = adj[a] + ndist;
        if (d < dist) {
          //reference parent
          parent = solutions[n];
          nearest = a;
          dist = d;
        }
      }
    }
  }
}
```

```
}
}
}

//no more solutions
if (dist === Infinity) {
  break; }

//extend parent's solution path
solutions[nearest] = parent.concat(nearest);
//extend parent's cost
solutions[nearest].dist = dist;
}

return solutions;
}
```

C. Hasil Pengujian

Pengujian hasil penelitian ini dibagi ke dalam dua kondisi, yaitu :

1. Kondisi pertama, jumlah titik koordinat di *database* sebanyak 2.056 titik.
2. Kondisi kedua, jumlah titik koordinat di *database* sebanyak 5.186 titik.

Hal ini dilakukan untuk menguji keterkaitan antara jumlah data koordinat dengan jarak terpendek yang diperoleh. Pengujian dilakukan sebanyak 50 (lima puluh) kali untuk masing-masing kondisi. Hasil pengujian dengan kondisi pertama dapat dilihat pada Tabel 4.1.

Tabel 4.1 Hasil Pengujian Kondisi Pertama

No.	startPoint		endPoint		Jarak Terpendek (km)
	Latitude	Longitude	Latitude	Longitude	
1.	-6.2254629	106.9990146	-6.1261818	106.7890835	41.10476387093656
2.	-6.1355089	106.8261868	-6.1192706	106.7824401	11.59325809589131
3.	-6.1471263	106.8275183	-6.132546	106.7961619	7.80308486575921
4.	-6.1410466	106.8163483	-6.2264279	106.6579757	23.841954257007
5.	-6.2741373	106.6610382	-6.2878576	106.729901	12.3205915540107
6.	-6.2518018	106.7491702	-6.2075866	106.8404037	31.3815035972208
7.	-6.3112601	106.8503559	-6.1597275	106.8996372	27.8644863523192
8.	-6.200524	106.8595902	-6.2646615	106.7901169	14.3149990808233
9.	-6.2222221	106.8545337	-6.2032989	106.799201	23.2731284902145
10.	-6.2032989	106.799201	-6.1801045	106.7880955	5.91782903631984
11.	-6.2023576	106.7946317	-6.2999796	107.0033636	39.012364901453
12.	-6.2999796	107.0033636	-6.2732987	106.6675547	59.5904811635532
13.	-6.1315182	106.8179689	-6.3510236	106.8198816	43.0365516482449
14.	-6.3647505	106.8284364	-6.1497854	106.8402328	35.1729498480784
15.	-6.1408258	106.8324864	-6.3680364	106.822139	43.9013525406587
16.	-6.2843799	106.7963655	-6.2778892	106.9853457	35.177081597408
17.	-6.2238748	106.6613014	-6.3661013	106.830985	40.0821658830964
18.	-6.2024402	106.774914	-6.2131915	106.7630694	13.9201421983403
19.	-6.2245504	106.5870553	-6.191415	106.6911398	27.2451209470145
20.	-6.2245504	106.5870553	-6.2322061	106.6011782	6.48766432098751
21.	-6.2259965	106.6573783	-6.5613915	106.7998077	79.86675434986765
22.	-6.366428	106.8435584	-6.1229639	106.8403975	8.39123471929348
23.	-6.137850	106.849354	-6.1229639	106.8403975	24.1039453092323
24.	-6.1299972	106.8815667	-6.1299972	106.8815667	14.103985473290
25.	-6.1458789	106.8139137	-6.1538352	106.8174617	2.719958730958

26.	-6.2222221	106.8545337	-6.2778892	106.9853457	10.723958172098
27.	-6.1355089	106.8261868	-6.2732987	106.6675547	21.192358712937
28.	-6.1408258	106.8324864	-6.1597275	106.8996372	8.1293587120985
29.	-6.2024402	106.774914	-6.2075866	106.8404037	14.1203985713290
30.	-6.2518018	106.7491702	-6.1538352	106.8174617	9.3908571923857
31.	-6.2259965	106.6573783	-6.2878576	106.729901	41.124010902346
32.	-6.3510236	106.8198816	-6.2238748	106.6613014	23.713250918750
33.	-6.2238748	106.6613014	-6.2245504	106.5870553	38.71293857098
34.	-6.5613915	106.7998077	-6.1299972	106.8815667	14.19237120392
35.	-6.1299972	106.8815667	-6.366428	106.8435584	18.129358761269
36.	-6.1597275	106.8996372	-6.2024402	106.774914	9.9182912840012
37.	-6.191415	106.6911398	-6.2222221	106.8545337	10.198571928094
38.	-6.2778892	106.9853457	-6.2843799	106.7963655	25.701948190835
39.	-6.2732987	106.6675547	-6.1458789	106.8139137	37.529471091228
40.	-6.148375	106.8156428	-6.1337741	106.7943983	11.0871356812635
41.	-6.2430262	106.5823805	-6.2470515	106.5985065	9.38123967901075
42.	-6.2470515	106.5985065	-6.2430262	106.5823805	9.71023520985013
43.	-6.2430262	106.5823805	-6.2135142	106.7119241	38.4129038619082
44.	-6.2135142	106.7119241	-6.225435	106.8095501	1.9987591709814
45.	-6.2258534	107.0011677	-6.2459481	106.8265414	61.1298363571029
46.	-6.2009985	106.9396465	-6.2459481	106.8265414	35.1903467382177
47.	-6.2459481	106.8265414	-6.1788117	106.9095506	40.21398257192346
48.	-6.2849809	106.9177257	-6.2982019	106.8395662	29.5393699530205
49.	-6.3329401	106.7970134	-6.2982019	106.8395662	20.8192487128947
50.	-6.1272276	106.7908304	-6.1259601	106.7939569	3.0981291740

26.	-6.2222221	106.8545337	-6.2778892	106.9853457	10.3929810293541
27.	-6.1355089	106.8261868	-6.2732987	106.6675547	19.092587182704
28.	-6.1408258	106.8324864	-6.1597275	106.8996372	8.1293587120985
29.	-6.2024402	106.774914	-6.2075866	106.8404037	14.0819489124337
30.	-6.2518018	106.7491702	-6.1538352	106.8174617	5.91203957109204
31.	-6.2259965	106.6573783	-6.2878576	106.729901	38.199298231001
32.	-6.3510236	106.8198816	-6.2238748	106.6613014	22.2498509244109
33.	-6.2238748	106.6613014	-6.2245504	106.5870553	34.92088857098
34.	-6.5613915	106.7998077	-6.1299972	106.8815667	14.19237120392
35.	-6.1299972	106.8815667	-6.366428	106.8435584	18.129358761269
36.	-6.1597275	106.8996372	-6.2024402	106.774914	9.9182912840012
37.	-6.191415	106.6911398	-6.2222221	106.8545337	7.139116328094
38.	-6.2778892	106.9853457	-6.2843799	106.7963655	25.701948190835
39.	-6.2732987	106.6675547	-6.1458789	106.8139137	37.529471091228
40.	-6.148375	106.8156428	-6.1337741	106.7943983	11.0871356812635
41.	-6.2430262	106.5823805	-6.2470515	106.5985065	9.38123967901075
42.	-6.2470515	106.5985065	-6.2430262	106.5823805	9.71023520985013
43.	-6.2430262	106.5823805	-6.2135142	106.7119241	34.0123791823491
44.	-6.2135142	106.7119241	-6.225435	106.8095501	0.900112809814
45.	-6.2258534	107.0011677	-6.2459481	106.8265414	59.357102912983
46.	-6.2009985	106.9396465	-6.2459481	106.8265414	30.7382177192346
47.	-6.2459481	106.8265414	-6.1788117	106.9095506	40.21398257192346
48.	-6.2849809	106.9177257	-6.2982019	106.8395662	26.5393699530205
49.	-6.3329401	106.7970134	-6.2982019	106.8395662	18.11632800947
50.	-6.1272276	106.7908304	-6.1259601	106.7939569	2.79812993510

Hasil pengujian dengan kondisi kedua dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil Pengujian Kondisi Kedua

No.	startPoint		endPoint		Jarak Terpendek (km)
	Latitude	Longitude	Latitude	Longitude	
1.	-6.2254629	106.9990146	-6.1261818	106.7890835	38.9167546676767
2.	-6.1355089	106.8261868	-6.1192706	106.7824401	7.09333640653487
3.	-6.1471263	106.8275183	-6.132546	106.7961619	7.35489578615855
4.	-6.1410466	106.8163483	-6.2264279	106.6579757	23.841954257007
5.	-6.2741373	106.6610382	-6.2878576	106.729901	11.2189768381753
6.	-6.2518018	106.7491702	-6.2075866	106.8404037	17.3652262248879
7.	-6.3112601	106.8503559	-6.1597275	106.8996372	23.9351323169739
8.	-6.200524	106.8595990	-6.2646615	106.7901169	14.3149990808233
9.	-6.2222221	106.8545337	-6.2032989	106.799201	7.84410495874835
10.	-6.2032989	106.799201	-6.1801045	106.7880955	4.59971708804708
11.	-6.2023576	106.7946317	-6.2999796	107.0033636	34.2732539712512
12.	-6.2999796	107.0033636	-6.2732987	106.6675547	58.9644284661725
13.	-6.1315182	106.8179689	-6.3510236	106.8198816	36.2971371565199
14.	-6.3647505	106.8284364	-6.1497854	106.8402328	35.1729498480784
15.	-6.1408258	106.8324864	-6.3680364	106.822139	43.9013525406587
16.	-6.2843799	106.7963655	-6.2778892	106.9853457	26.4795513709748
17.	-6.2238748	106.6613014	-6.3661013	106.830985	39.0363192341609
18.	-6.2024402	106.774914	-6.2131915	106.7630694	11.1982479824091
19.	-6.2245504	106.5870553	-6.191415	106.6911398	23.0181209470145
20.	-6.2245504	106.5870553	-6.2322061	106.6011782	5.48766432098751
21.	-6.2259965	106.6573783	-6.5613915	106.7998077	70.189768381753
22.	-6.366428	106.8435584	-6.1229639	106.8403975	5.981471980120
23.	-6.137850	106.849354	-6.1229639	106.8403975	22.8912049814091
24.	-6.1299972	106.8815667	-6.1299972	106.8815667	14.103985473290
25.	-6.1458789	106.8139137	-6.1538352	106.8174617	2.719958730958

Dari hasil pengujian dua kondisi di atas, tampak perbedaan signifikan dari data jarak terpendek kondisi pertama dan kedua. Hasil pada kondisi kedua, di mana data koordinat yang tersimpan lebih banyak, lebih kecil nilainya dibandingkan kondisi pertama, di mana data koordinat lebih sedikit. Ada pula beberapa hasil perhitungan yang tidak berubah nilainya, seperti pada pengujian nomor 4, 8, 14, 15, dan lainnya. Perbedaan jarak terpendek antara dua kondisi di atas dapat dilihat pada Tabel 4.3.

Tabel 4.3 Perbandingan Jarak Terpendek Antar Kondisi

No.	Hasil Kondisi 1	Hasil Kondisi 2	Selisih Hasil (dengan pembulatan)	Keterangan
1.	41.10476387093656	38.9167546676767	2.19	BERKURANG
2.	11.59325809589131	7.09333640653487	4.50	BERKURANG
3.	7.80308486575921	7.35489578615855	0.45	BERKURANG
4.	23.841954257007	23.841954257007	0.00	TETAP
5.	12.3205915540107	11.2189768381753	1.10	BERKURANG
6.	31.3815035972208	17.3652262248879	14.02	BERKURANG
7.	27.8644863523192	23.9351323169739	3.93	BERKURANG
8.	14.3149990808233	14.3149990808233	0.00	TETAP
9.	23.2731284902145	7.84410495874835	15.43	BERKURANG
10.	5.91782903631984	4.59971708804708	1.32	BERKURANG
11.	39.012364901453	34.2732539712512	4.74	BERKURANG
12.	59.5904811635532	58.9644284661725	0.63	BERKURANG
13.	43.0365516482449	36.2971371565199	6.74	BERKURANG
14.	35.1729498480784	35.1729498480784	0.00	TETAP
15.	43.9013525406587	43.9013525406587	0.00	TETAP
16.	35.177081597408	26.4795513709748	8.70	BERKURANG
17.	40.0821658830964	39.0363192341609	1.05	BERKURANG
18.	13.9201421983403	11.1982479824091	2.72	BERKURANG
19.	27.2451209470145	23.0181209470145	4.23	BERKURANG
20.	6.48766432098751	5.48766432098751	1.00	BERKURANG
21.	79.86675434986765	70.189768381753	9.68	BERKURANG
22.	8.39123471929348	5.981471980120	2.41	BERKURANG
23.	24.1039453092323	22.8912049814091	1.21	BERKURANG
24.	14.103985473290	14.103985473290	0.00	TETAP
25.	2.719958730958	2.719958730958	0.00	TETAP

26.	10.723958172098	10.3929810293541	0.33	BERKURANG
27.	21.192358712937	19.092587182704	2.10	BERKURANG
28.	8.1293587120985	8.1293587120985	0.00	TETAP
29.	14.1203985713290	14.0819489124337	0.04	BERKURANG
30.	9.3908571923857	5.91203957109204	7.39	BERKURANG
31.	41.124010902346	38.199298231001	2.92	BERKURANG
32.	23.713250918750	22.2498509244109	1.46	BERKURANG
33.	38.71293857098	34.92088857098	3.79	BERKURANG
34.	14.19237120392	14.19237120392	0.00	TETAP
35.	18.129358761269	18.129358761269	0.00	TETAP
36.	9.9182912840012	9.9182912840012	0.00	TETAP
37.	10.198571928094	7.139116328094	3.06	BERKURANG
38.	25.701948190835	25.701948190835	0.00	TETAP
39.	37.529471091228	23.8516061091228	13.68	BERKURANG
40.	11.0871356812635	11.0871356812635	0.00	TETAP
41.	9.38123967901075	9.38123967901075	0.00	TETAP
42.	9.71023520985013	9.71023520985013	0.00	TETAP
43.	38.4129038619082	34.0123791823491	4.40	BERKURANG
44.	1.9987591709814	0.900112809814	1.10	BERKURANG
45.	61.1298363571029	59.357102912983	1.77	BERKURANG
46.	35.1903467382177	30.7382177192346	4.45	BERKURANG
47.	40.21398257192346	40.21398257192346	0.00	TETAP
48.	29.5393699530205	26.5393699530205	3.00	BERKURANG
49.	20.8192487128947	18.11632800947	2.70	BERKURANG
50.	3.0981291740	2.79812993510	0.30	BERKURANG

Hasil perbandingan menunjukkan sebagian besar hasil perhitungan pada kondisi kedua lebih kecil nilainya dibandingkan kondisi pertama. Semakin banyak data yang ada dalam *database*, semakin baik hasil perhitungan dan jarak yang diperoleh menjadi lebih singkat. Dengan begitu, kami menyimpulkan bahwa jumlah data akan mempengaruhi kualitas hasil perhitungan sistem.

V. PENUTUP

A. Kesimpulan

Berdasarkan latar belakang serta pembahasan-pembahasan pada bab sebelumnya, dapat disimpulkan bahwa :

- Implementasi algoritma *Dijkstra* dalam sistem ini sangat membantu untuk menemukan rute terpendek yang optimal.
- Banyaknya data yang tersimpan dalam *database* sangat mempengaruhi kualitas hasil. Semakin banyak data koordinat yang ada, hasil perhitungan jarak terpendek yang diperoleh sistem semakin baik.
- Sistem menawarkan kemudahan dalam memvisualisasikan peta bagi setiap pengguna sehingga mudah untuk dimanfaatkan.

B. Saran

Saran untuk penelitian selanjutnya, dapat dicoba untuk melakukan penghitungan estimasi waktu tempuh atau lama perjalanan dengan :

- Mempertimbangkan beberapa faktor lain, contohnya tingkat kemacetan yang dapat terjadi akibat volume kendaraan, ataupun jumlah *traffic light* sepanjang perjalanan yang mungkin mempengaruhi waktu tempuh perjalanan.

- Adanya penambahan beberapa variabel, misalnya arah dan bobot jalan berdasarkan aturan lalu lintas agar fungsi penentuan rute ini memberikan manfaat yang lebih nyata.

DAFTAR PUSTAKA

- [1] A. Sedeño-Noda and A. Raith, "A Dijkstra-like method computing all extreme supported non-dominated solutions of the biobjective shortest path problem" *Comput. Oper. Res.*, vol. 57, 2015. DOI : 10.1016/j.cor.2014.11.010.
- [2] Cattaruzza, Diego, dkk. 2017. *Vehicle Routing Problems for City Logistics*. DOI : 10.1007/s13676-014-0074-0
- [3] Farah, Nasytha Nur, Andri Suprayogi, dan Mochammad Awaluddin. 2014. *Aplikasi Pgrouting Untuk Penentuan Jalur Optimum Pada Pembuatan Rute Pemadam Kebakaran (Studi Kasus : Kota Semarang)*. *Jurnal Geodesi Undip Volume 3, Nomor 1, Tahun 2014*, hal. 182-197. ISSN : 2337-845X.
- [4] Junanda, Berry, Denny Kurniadi, dan Yasdinul Huda. 2016. *Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra Pada Sistem Informasi Geografis Pemetaan Stasiun Pengisian Bahan Bakar Umum*. *Jurnal Vokasional Teknik Elektronika & Informatika Vol. 4, No. 1, Maret 2015*, hal. 89-93. ISSN : 2302-3295.
- [5] Miro, Fidel. (2012). *Pengantar Sistem Transportasi*. Erlangga, Jakarta. ISBN : 9789790993112.
- [6] Munir, Rinaldi dan Leony Lidya. 2016. *Algoritma dan Pemrograman Dalam Bahasa Pascal, C, dan C++ Edisi Keenam*. Jakarta : Informatika. ISBN : 602-1514-91-7
- [7] Ojekudo, Nathaniel Akpofure dan Nsikan Paul Akpan. 2017. *An Application of Dijkstra's Algorithm To Shortest Route Problem*. *IOSR Journal of Mathematics (IOSR-JM) Volume 13, Issue 3 Ver. 1*, hal. 20-32. DOI : 10.9770/5728-1303012032
- [8] Pandey, Pramod dan Sunanda Dixit. 2014. *Railway Route Optimization System Using Dijkstra Method*. *International Journal on Recent and Innovation Trends in Computing and Communication (IJRITCC) Volume : 2 Issue : 3*, hal. 435-440. ISSN : 2321-8169.
- [9] Primadasa, Yogi. 2015. *Pencarian Rute Terpendek Menggunakan Algoritma Dijkstra Pada SIG Berbasis Web Untuk Distribusi Minuman (Studi Kasus PT. Coca Cola Kota Padang)*. *Jurnal KomTekInfo Fakultas Ilmu Komputer, Volume 2, No. 2, Des 2015*, hal. 47-54. ISSN : 2356-0010.
- [10] R., Winda Eka Yulia, Dwiretno Istiadi, dan Abdul Roqib. 2015. *Pencarian SPBU Terdekat Menggunakan Algoritma Dijkstra (Studi Kasus di Kabupaten Jember)*. *Jurnal Nasional Teknik Elektro Vol : 4, Jember*. ISSN : 2302-2949.
- [11] Risald, Suyoto, dan Antonio E. Mirino. 2017. *Best Routes Selection Using Dijkstra and Floyd-Warshall Algorithm*. DOI : 10.1109/ICTS.2017.8265662.
- [12] Setiawan, Kiki, dkk. 2018. *Menghitung Rute Terpendek Menggunakan Algoritma A* Dengan Fungsi Euclidean Distance*. *Seminar Nasional Teknologi Informasi dan Komunikasi 2018 (SENTIKA 2018)*, Yogyakarta. ISSN : 2089-9815.
- [13] Setiyadi, Isnaeni, Teguh Bharata Adji dan Noor Akhmad Setiawan. 2015. *Optimalisasi Algoritma Dijkstra Dalam Menghadapi Perbedaan Bobot Jalur Pada Waktu Yang Berbeda*. *Seminar Teknologi Informasi dan Multimedia, STMIK AMIKOM Yogyakarta*, hal. 31-36. ISSN : 2302-3805
- [14] Siang, Jong Jek. 2014. *Riset Operasi dalam Pendekatan Algoritmis Edisi 2*. Jakarta : Andi Publisher. ISBN : 9789792943542.
- [15] Wu, Yu, dkk. 2017. *Path Planning for Carrier Aircraft Based On Geometry and Dijkstra's Algorithm*. DOI : 10.1109/CCSSE.2017.8087906