ISSN : 1411-1624
e-ISSN : 2621-8089

PENENTUAN RUTE TERPENDEK MENUJU PUSAT PERBELANJAAN DI JAKARTA MENGGUNAKAN ALGORITMA DIJKSTRA

Jodi Setiawan[1], Rezy S. Prakoso[2], Kristien Margi Suryaningrum[3]
Mahasiswa Universitas Bunda Mulia[1,2], Dosen Universitas Bunda Mulia[3]
Jalan Lodan Raya, RT.12/RW.2, Ancol, Kec. Pademangan, Kota Jakarta Utara, Daerah Khusus Ibukota Jakarta
Sur-el : jodi.set2106@gmail.com[1], reji0599@gmail.com[2], kristienmargi@gmail.com[3]

Abstract : *Shopping centers are easy to find in big cities like Jakarta and because there are too many times people are often confused which one to go to, not to mention if people from outside Jakarta come and want to visit a shopping center. The problem to choose a shopping center that encourages researchers to conduct a study to develop applications to find the nearest shopping center using the Dijkstra algorithm that can be used on smartphones in order to make it easier to find and find shopping centers around users. The application was designed using flowchart and UML and was made using Android Studio with the help of the Google Maps API to display maps, display pins, find and display the nearest route using latitude and longitude from Google Maps. Algorithm to calculate and determine the route that must be traversed using the Dijkstra algorithm. The accuracy of the calculation results using the Dijkstra algorithm reaches 93%.*

Keywords: *Dijkstra, Google Maps API, Android Studio, Graph*

Abstrak : *Pusat perbelanjaan mudah ditemui di kota-kota besar seperti di Jakarta dan karena jumlahnya yang terlalu banyak sering kali orang-orang bingung untuk pergi ke pusat perbelanjaan yang mana, belum lagi jika orang-orang dari luar Jakarta datang dan ingin mengunjungi suatu pusat perbelanjaan. Permasalahan untuk memilih pusat perbelanjaan itu yang mendorong peneliti melakukan sebuah penelitian untuk mengembangkan aplikasi untuk mencari pusat perbelanjaan terdekat dengan menggunakan algoritma Dijkstra yang dapat digunakan pada smartphone agar memudahkan untuk mengetahui dan menemukan pusat perbelanjaan yang ada di sekitar pengguna. Aplikasi dirancang menggunakan flowchart dan UML dan dibuat dengan menggunakan Android Studio dengan bantuan Google Maps API untuk menampilkan peta, menampilkan pin, mencari dan menampilkan rute terdekatnya menggunakan latitude dan longitude dari Google Maps. Algoritma untuk menghitung dan menentukan rute yang harus dilalui menggunakan algoritma Dijkstra. Keakuratan hasil hitung dengan menggunakan algoritma Dijkstra mencapai 93%.***

Kata kunci*: Dijkstra, Google Maps API, Android Studio, Graph*

1. PENDAHULUAN

Pusat perbelanjaan adalah sekelompok usaha ritel dan usaha komersial lainnya yang direncanakan, dikembangkan, dimiliki, dan dikelola sebagai satu properti tunggal [1]. Pusat perbelanjaan dapat ditemui di kota, terutama kota-kota besar seperti di Jakarta. Banyak orang berbondong-bondong datang setiap harinya ke pusat perbelanjaan dari berbagai rentang usia, mulai dari remaja hingga orang dewasa. Sistem belanja *online* akhir-akhir ini lebih digemari namun tetap saja beberapa orang lainnya lebih memilih untuk berbelanja secara langsung ke pusat perbelanjaan. Alasan-alasan yang mendukungnya seperti dikarenakan berbelanja secara langsung akan lebih mudah mengetahui bagaimana kualitas dan bentuk barang yang akan dibeli secara langsung, selain itu juga belanja

ISSN : 1411-1624
e-ISSN : 2621-8089

secara langsung tidak dikenakan biaya pengiriman dan barang yang dibeli dapat didapatkan di hari itu juga [2]. Itulah beberapa alasan mengapa berbelanja secara langsung masih digemari sampai saat ini walaupun tren belanja *online* sedang ramai-ramainya.

Pusat perbelanjaan pun sekarang ini telah mengalami perubahan fungsi. Dahulu aktivitas di pusat perbelanjaan hanya sebatas berbelanja namun sekarang pusat perbelanjaan sudah difungsikan lebih dari itu. Beberapa orang sekarang ini menggunakan pusat perbelanjaan sebagai tempat untuk menghabiskan waktu, berkumpul bersama, sebagai tempat hiburan, dan mengadakan suatu acara. Orang-orang dari luar daerah akan sulit menemukan pusat perbelanjaan yang ada di Jakarta karena jumlahnya yang sangat banyak. Permasalahan untuk mencari rute inilah yang membuat peneliti membuat sebuah aplikasi yang dapat membantu orang-orang mencari pusat perbelanjaan di Jakarta.

2. METODOLOGI PENELITIAN

Metode pengembangan sistem yang penulis gunakan pada penelitian ini yaitu metode *Waterfall* Model [3]. Metode ini terdiri dari beberapa tahapan yaitu:

1. Tahap *requirement*, tahap ini akan direncanakan apa saja kebutuhan perangkat yang akan diinginkan oleh pengguna.
2. Tahap *design*, pada tahap ini direncanakan bagaimana sistem bekerja,
3. Tahap *implementation*, pada tahap ini penulis akan mulai memangun program yang telah direncanakan atas tahap-tahap berikutnya.
4. Tahap *verification*, pada tahap ini sistem yang dibuat akan diuji coba dan dicari kesalahannya.
5. Tahap *maintenance*, pada tahap ini sistem diluncurkan dan dirawat.

2.1 *Dijkstra*

Algoritma Dijkstra adalah sebuah algoritma yang ditemukan oleh Edsger W [4]. Dijkstra yang menggunakan prinsip algoritma rakus (*greedy*) untuk menentukan rute terpendeknya dalam sebuah graf berarah dengan bobot-bobot sisi yang tidak bernilai negatif, namun hal ini juga berlaku pada graf tak berarah. Prinsip dari *greedy* pada algoritma Dijkstra menyatakan bahwa pada setiap langkah dipilih sisi yang berbobot minimum dan memasukkannya dalam himpunan solusi. Input algoritma ini adalah sebuah graf G yang berbobot (*weighted graph*) dan sebuah titik sumber, dengan V adalah himpunan semua titik (*vertex*) dan E himpunan semua sisi dalam graf G. Setiap sisi dari graf ini adalah pasangan titik (u,v) yang melambangkan hubungan dari titik u ke titik v. Bobot (*weight*) dari sebuah sisi dapat dianggap sebagai jarak antara dua titik, yaitu panjang sisi dari dua titik [5]. Proses pencarian pusat perbelanjaan terdekat dijelaskan sebagai berikut.

```
FOR i ← 0 TO V DO
      distance[i] ← infinite
      vert_visit[i] ← false
END FOR
parents[vert_start] ← -1
distance[vert_start] ← 0
FOR i ← 1 TO V DO
      min ← infinite
      min_index ← -1
```

ISSN : 1411-1624
e-ISSN : 2621-8089

```
      FOR n ← 0 TO V THEN
      IF vert_visit[n] = false AND
      distance[n] < min THEN
            min ← distance[n]
            min_index ← n
      END IF
END FOR
vert_visit[min_index] ← true
FOR n ← 0 TO V DO
      temp_sum ← G[min_index][n] +
      min
      IF G[min_index][n] !=0 AND
      vert_visit[n] = false AND
      temp_sum != infinite AND
      temp_sum <distance[n] THEN
            parents[n] ← min_index
            distance[n] ← temp_sum
      END IF
END FOR
```

2.2. *Android Studio*

Android Studio merupakan IDE resmi untuk pengembangan aplikasi *Android* dan bersifat *open source* atau gratis [6]. *Android Studio* memiliki fitur editor kode cerdas (*Intelligent Code Editor*) yang memiliki kemampuan penyelesaian kode, optimalisasi, dan analisis kode yang canggih [7].

2.3. *Google Maps API*

Google Maps adalah layanan aplikasi peta *online* yang disediakan oleh *Google* secara gratis. Layanan peta *Google Maps* secara resmi dapat diakses melalui situs *http://maps.google.com*. Pada situs tersebut dapat dilihat informasi geografis pada hampir semua permukaan di bumi kecuali daerah kutub utara dan selatan [8].

Google Maps API merupakan perkembangan dari *google Maps*. Dengan menggunakan *google Maps API* ini, dimungkinkan untuk dapat menggunakan *google Maps* di dalam *website*. Meski awalnya hanya *JavaScript API*, *Maps API* diperluas untuk menyertakan sebuah *API* untuk aplikasi *Adobe Flash*. Keberhasilan *google Maps API* telah melahirkan sejumlah pesaing antara lain *Yahoo! Maps API*, *Bing Maps Platforn*, *MapQuest Development Platform* dan *OpenLayers* [9].

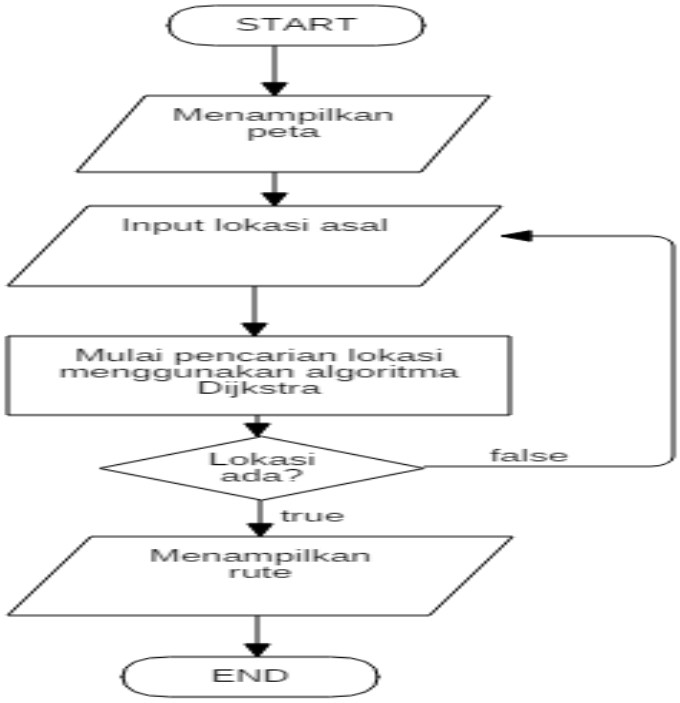
2.3. *Unified Modelling Language*

Unified Modelling Language (UML) adalah suatu notasi pendiagraman yang tepat yang mampu mengijinkan suatu desain program untuk direpresentasikan dan didiskusikan [10]. *UML* akan digunakan untuk menjelaskan gambaran dari sistem yang dibuat.

3. HASIL DAN PEMBAHASAN

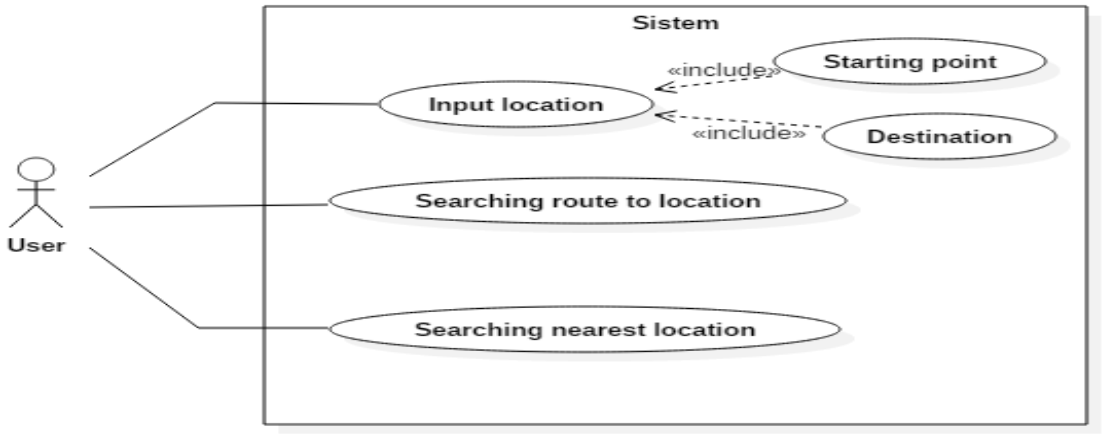
Analisis kebutuhan mencakup kebutuhan-kebutuhan yang harus dapat dilakukan oleh sistem dalam menjawab kebutuhan dari pengguna. Aplikasi yang dirancang harus dapat menerima lokasi yang dimasukkan oleh pengguna. Ketika pengguna telah selesai memasukkan alamat dan mulai menekan tombol cari yang ada di aplikasi sistem harus dapat mencari dan menampilkan hasil yang diinginkan oleh pengguna. Ada juga fitur untuk mencari lokasi pusat perbelanjaan yang terdekat dari posisi pengguna yang diambil dari *GPS*. Alur kerja aplikasi dijelaskan oleh *flowchart* seperti pada gambar 1.

ISSN : 1411-1624
e-ISSN : 2621-8089



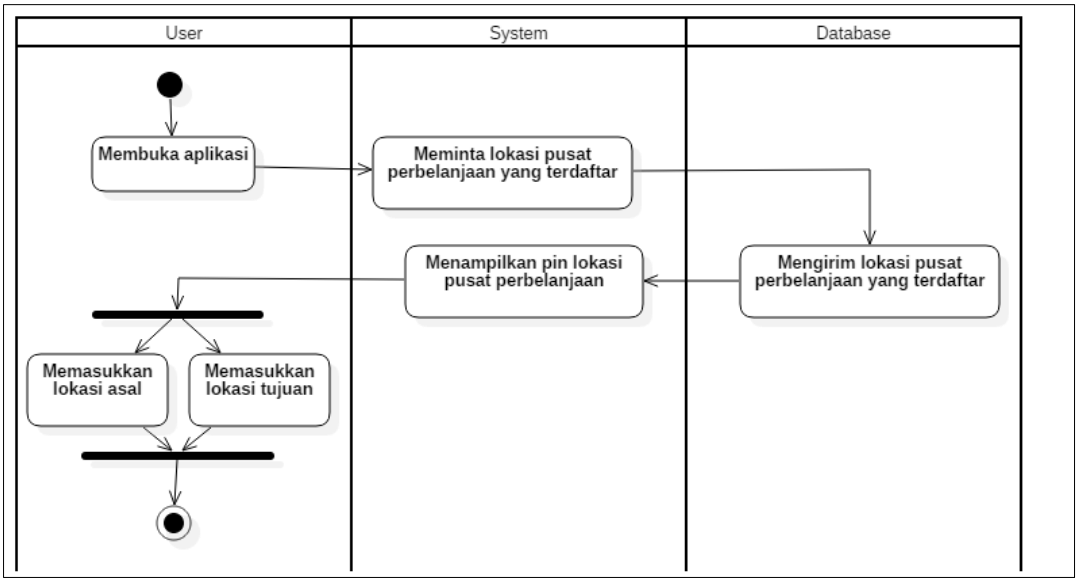
Gambar 1. *Flowchart* aplikasi

Penggambaran sistem lebih dijelaskan menggunakan *UML* dan terdiri dari empat macam diagram.*Use case diagram* seperti yang terlihat pada gambar 2 digunakan untuk menggambarkan hubungan antara fungsi yang tersedia di dalam sistem dan yang dapat digunakan oleh aktor atau *user*. Pada aplikas hanya ada satu aktor yaitu *user* dan tiga buah *use case*, yaitu *Input location* dimana *user* dapat memasukkan lokasi asal dan memilih lokasi tujuan yang disediakan oleh sistem, *searching route to location* dimana *user* dapat mendapatkan informasi menuju pusat perbelanjaan terdekat, dan *searching nearest location* dimana *user* dapat mendapatkan daftar pusat perbelanjaan terdekat dari posisi sekarang.

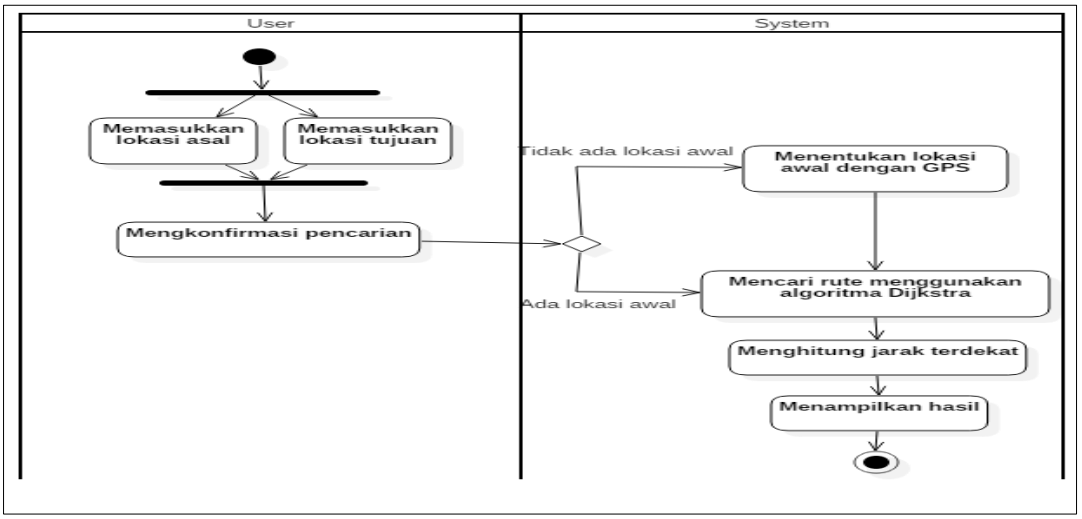


Gambar 2. *Use Case Diagram*

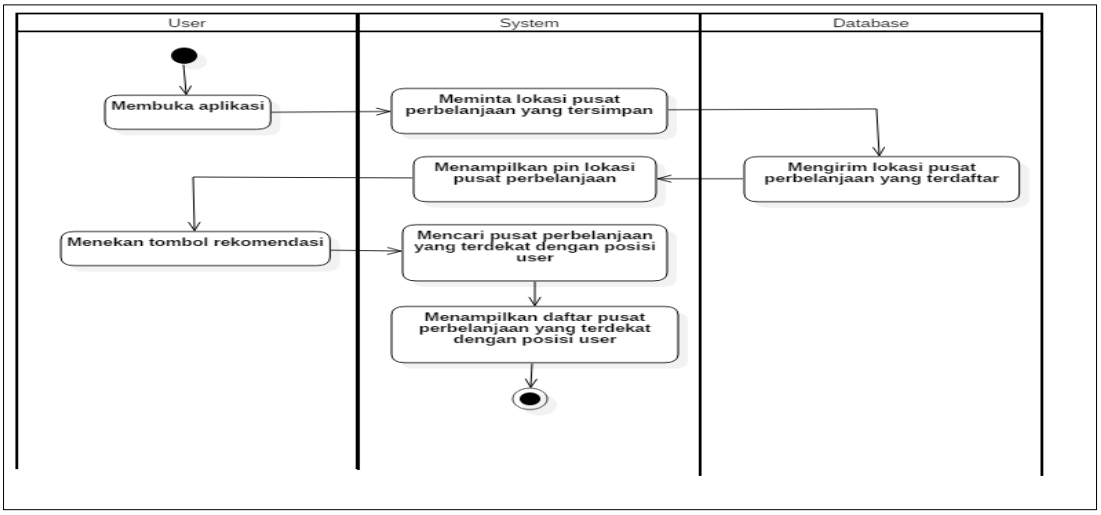
ISSN : 1411-1624
e-ISSN : 2621-8089



Gambar 3. ***Activity input location***



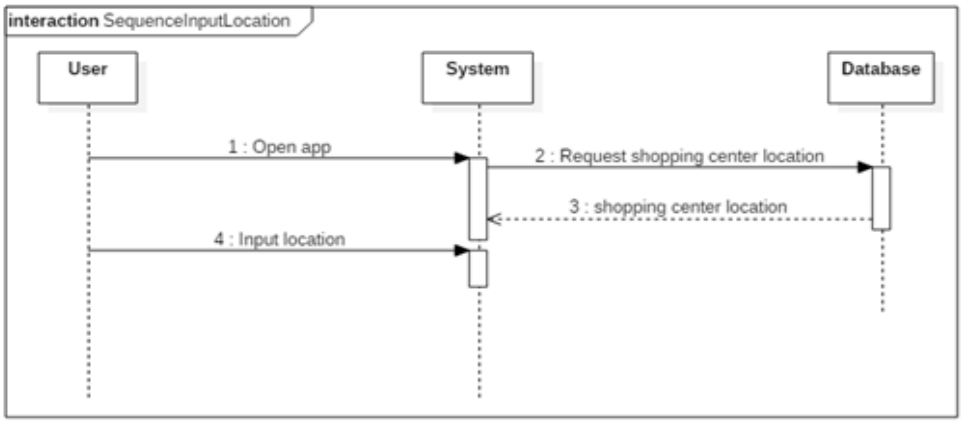
Gambar 4. ***Activity searching route to location***



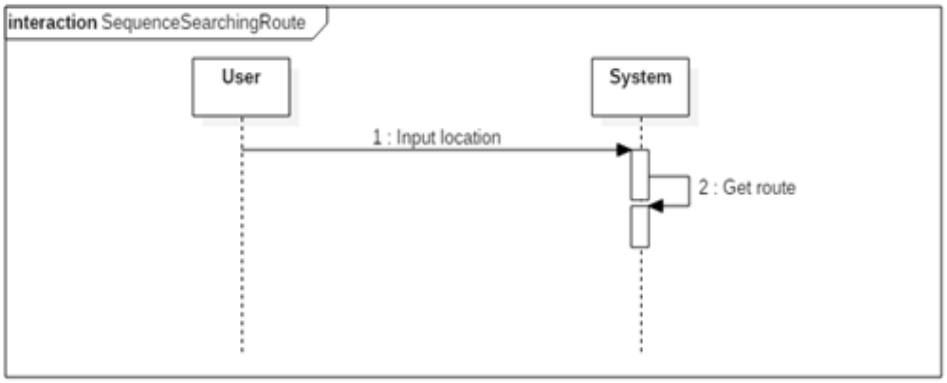
Gambar 5. ***Activity searching nearest location***

ISSN : 1411-1624
e-ISSN : 2621-8089

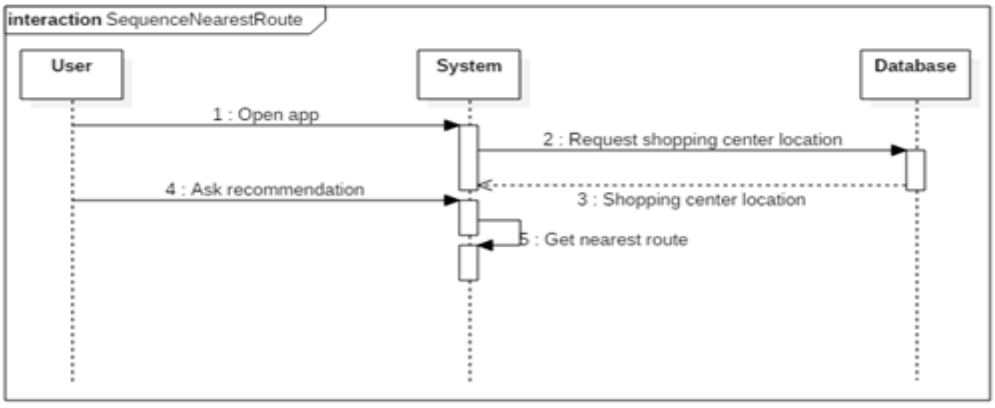
Activity diagram digambarkan seperti pada gambar 3, 4, dan 5. *Activity input location* pada gambar 3 menjelaskan setiap proses aktivitas yang perlu dieksekusi untuk menjalankan proses memasukkan lokasi yang dilakukan oleh *user* kepada sistem. *Activity searching route to location* pada gambar 4 menjelaskan alur proses aktivitas untuk mengeksekusi pencarian menemukan rute ke lokasi yang diinginkan oleh *user*. *Activity searching nearest location* pada gambar 5 adalah fitur tambahan yang disediakan di dalam aplikasi.



Gambar 6. ***Sequence input location***



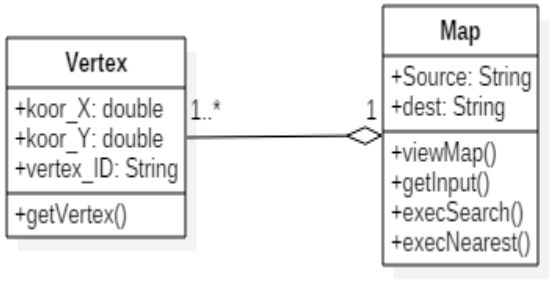
Gambar 7. ***Sequence searching route to location***



Gambar 8. ***Sequence searching nearest location***

ISSN : 1411-1624
e-ISSN : 2621-8089

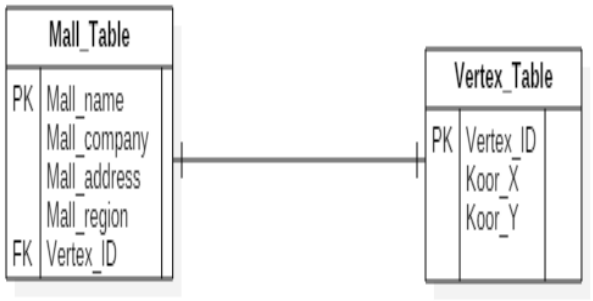
Sequence diagram digambarkan seperti pada gambar 6, 7, dan 8. *Sequence input location* pada gambar 6 menjelaskan tentang urutan pengiriman pesan yang terjadi saat *user* membuka dan memasukkan lokasi. *Sequence searching route to location* pada gambar 7 menjelaskan tentang pengiriman pesan di dalam sistem saat *user* meminta sistem untuk menampilkan rute menuju lokasi pusat perbelanjaan yang diinginkan oleh *user*. *Sequence searching to nearest location* pada gambar 8 menjelaskan alur pengiriman pesan di saat *user* meminta sistem untuk memberikan lokasi pusat perbelanjaan yang ada di sekitarnya.



Gambar 9. *Class diagram*

Class diagram seperti pada gambar 9 memiliki dua objek, yaitu objek Vertex dan Maps. Objek Vertex digunakan untuk menyimpan informasi tentang *vertex* atau titik yang nantinya akan ditampilkan pada peta. Objek Map adalah pusat dari operasi yang dikerjakan oleh sistem, mulai dari menampilkan peta, menerima *input*, sampai menghitung dan mencari rute terdekat dari *user*.

Perancangan *database* yang digunakan di dalam aplikasi yang dalam penelitian ini dirancang dengan menggunakan model *Entity Relationship Diagram* (*ERD*) dengan notasi *crowsfoot*, dimana *ERD* yang akan digunakan terlihat seperti pada gambar 10.

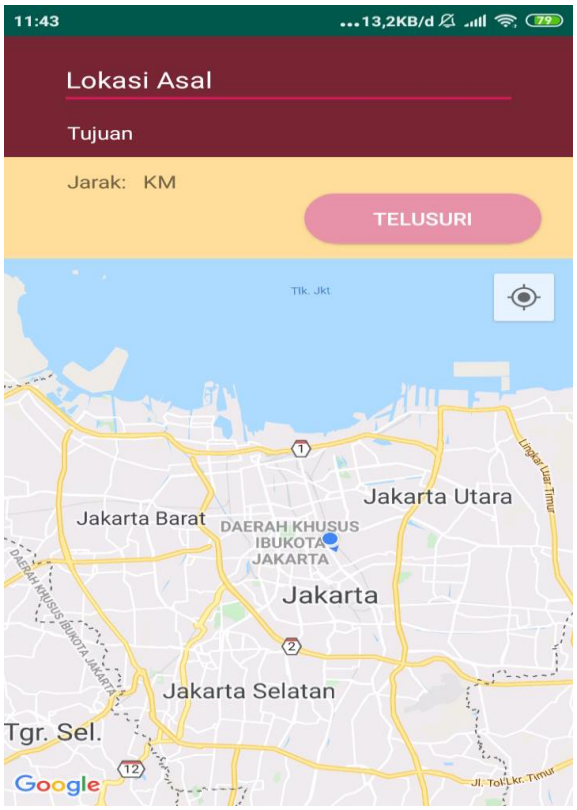


Gambar 10. *Entity relationship diagram*

Database terdiri dari dua tabel, yaitu tabel untuk vertex atau titik yang diberi nama Vertex_Table dan tabel untuk menyimpan informasi lokasi pusat perbelanjaan yang diberi nama Mall_Table. Tabel Mall_Table memiliki relasi satu ke satu (*one-to-one relationship*) dengan tabel Vertex_Table karena sebuah pusat perbelanjaan hanya bisa menempati satu tempat atau *vertex*, begitu juga sebaliknya diaman satu tempat atau *vertex* hanya bisa ditempati oleh satu pusat perbelanjaan.

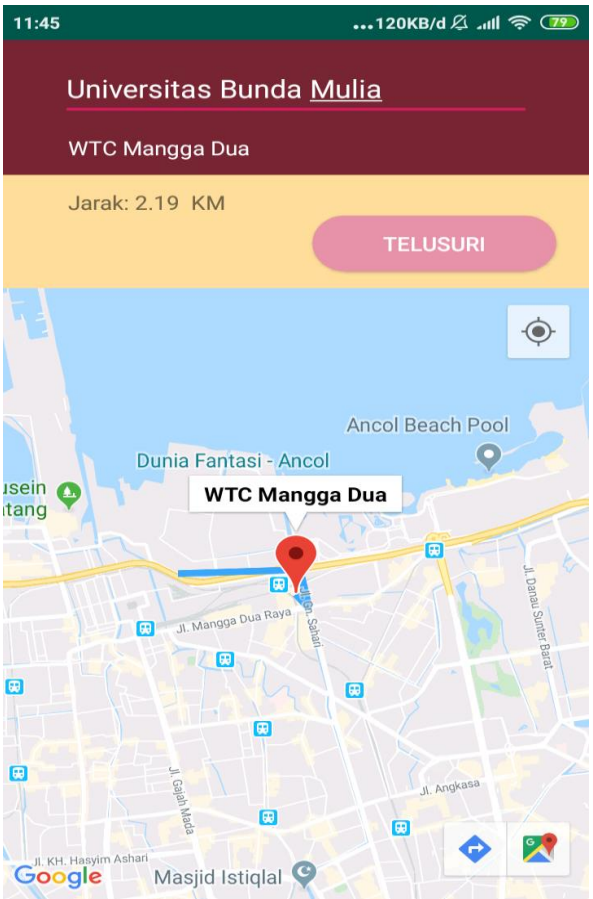
Tampilan aplikasi terlihat seperti pada gambar 11. Terdapat peta yang mendominasi sebagian layar lebih. Dua buah *label* untuk menunjukkan lokasi tujuan *user* dan untuk menunjukkan jarak. Sebuah *edit box* dimana user dapat memasukkan lokasi asal ataupun tidak (jika ingin menjadikan lokasi sekarang *user* sebagai lokasi asal). Sebuah *button* untuk memulai pencarian pusat perbelanjaan terdekat. Tiga buah *button* untuk menampilkan rekomendasi pusat perbelanjaan yang terdekat yang diatur tidak terlihat sampai *button* dengan *caption* ‘telusuri’ ditekan oleh *user*. Keluaran yang ditampilkan oleh aplikasi terlihat seperti pada gambar 12.

ISSN : 1411-1624
e-ISSN : 2621-8089



Gambar 11. Tampilan awal aplikasi

.



Gambar 12. Tampilan akhir aplikasi

Pengujian dilakukan dengan menggunakan sepuluh lokasi asal dan ketiga lokasi yang direkomendasikannya, menghasilkan total tiga puluh pengujian Hasil dari pengujian algoritma akan terlihat seperti pada tabel 1. Jarak yang didapatkan dari menghitung jarak terdekat antara kedua *vertex* yang saling bertetanggaan. Jarak tersebut diakumulasikan dari lokasi asal menuju lokasi tujuan baru dikatakan sebagai jarak dari titik asal menuju titik tujuan.

Pengujian yang dilakukan dalam tiga puluh percobaan menunjukkan terdapat 28 perhitungan dengan hasil yang valid dan dua perhitungan dengan hasil yang tidak valid. Salah satu hasil yang tidak valid terdapat saat dilakukan pengujian dari La Piazza menuju Mal Kelapa Gading dimana sistem menghitung jaraknya 0.9 KM sementara peneliti menghitungnya 0.91 KM. Hitungan tidak valid kedua ditemukan pada saat melakukan pengujian dari Mall of Indonesia menuju La Piazza dimana hasil dari sistem menunjukkan jarak sebesar 2.62 KM sementara dari hitungan peneliti adalah 2.67 KM. Persentase keberhasilan dilakukan dengan menggunakan rumus pada persamaan (1) dan menunjukkan tingkat keberhasilan sebesar 93%.

$$Tkt\ \ Keberhasilan = \frac{Pengujian\ yang\ Valid}{Totla\ Pengujian} \ldots\ldots\ldots\ldots (1)$$

ISSN : 1411-1624
e-ISSN : 2621-8089

Tabel 1. Hasil Pengujian

NO	*Lokasi Asal*	*Lokasi Tujuan*	*Jarak (KM)*	*Hitungan*	*Validasi*
1	Universitas Bunda Mulia	WTC Mangga Dua	2.19	2.19	Valid
		ITC Mangga Dua	2.54	2.54	Valid
		Mangga Dua Mal	2.8	2.8	Valid
2	Golden Truly	Haroo Pasar Baru	0.88	0.88	Valid
		Istana Pasar Baru	0.99	0.99	Valid
		Pusat Perbelanjaan MC	1.66	1.66	Valid
3	ITC Roxy Mas	Mal Ciputra	1.82	1.82	Valid
		Plaza Gajah Mada	2.46	2.46	Valid
		Central Park	2.82	2.82	Valid
4	Mal Cilandak	Cilandak Town Square	2.88	2.88	Valid
		Mall Pejaten Village	3.43	3.43	Valid
		Poins Square	5.88	5.88	Valid
5	La Piazza	Mal Kelapa Gading	0.9	0.91	Tidak Valid
		Mal of Indonesia	2.67	2.67	Valid
		Mal artha Gading	3.48	3.48	Valid
6	Mall Of Indonesia	Mall Artha Gading	1.71	1.71	Valid
		ITC Cempaka Mas Mega	2.62	2.62	Valid
		La Piazza	2.62	2.67	Tidak Valid
7	Plaza Atrium	Plaza Kenari Mas	1.89	1.89	Valid
		Menteng Huis	2.01	2.01	Valid
		Golden Truly	2.76	2.76	Valid
8	Grand Cakung	Pulo Gading Trade Centre	4.85	4.85	Valid
		Plaza Buaran	6.57	6.57	Valid
		Pokets	7.85	7.85	Valid
9	Grand Paragon	Plaza Gajah Mada	1.05	1.05	Valid
		Lindeteves trade Center	2.46	2.46	Valid
		Istana Pasar Baru	2.49	2.49	Valid
10	Mal Ciputra	Central Park	1.48	1.48	Valid
		Mal Taman Anggrek	1.59	1.59	Valid
		ITC Roxy Mas	1.82	1.82	Valid

4. KESIMPULAN

Kesimpulan dari penelitian yang dilakukan adalah algoritma Dijkstra dapat diimplementasikan untuk mencari jarak terpendek menuju pusat perbelanjaan terdekat dengan keakuratan 93% dengan perbandingan yang dilakukan antara hasil perhitungan yang dilakukan oleh sistem dan hasil perhitungan yang dilakukan oleh peneliti. Implementasi algoritma Dijkstra ke dalam *Google Maps API* juga dapat dilakukan jika menggunakan database *vertex* yang didata sendiri karena untuk implementasi menggunakan *vertex* dari *Google Maps*, dibutuhkan *key* yang harus diperoleh dengan membelinya.

DAFTAR PUSTAKA

[1] G. T. Sari, "Pusat perbelanjaan mall di kabupaten kubu raya," *Jurnal online mahasiswa Arsitektur Universitas Tanjungpura*, Vol. 5, no. 2 September 2017, pp. 1–12, 2017.

[2] U. Widowati, "10 Alasan Belanja Langsung Lebih Asyik dari Belanja Online," *CNN Indonesia*, 2015. [Online]. Available: https://www.cnnindonesia.com/gaya-hidup/20150729080646-277-68749/10-alasan-belanja-langsung-lebih-asyik-dari-belanja-online. [Accessed: 13-Sep-2019].

ISSN : 1411-1624
e-ISSN : 2621-8089

[3] M. Tabrani and P. Eni, "Penerapan Metode Waterfall Pada Sistem Informasi Inventori Pt. Pangan Sehat Sejahtera," *J. Inkofar*, vol. 1, no. 2, pp. 30–40, 2017.
[4] A. Zaki, "Algoritma Dijkstra : Teori Dan Aplikasinya," *J. Mat. UNAND*, vol. 6, no. 4, pp. 1–8, 2018.
[5] K. S. Zalukhu, "Implementasi Algoritma Dijkstra dalam Pencarian Rute Efektif Menghindari Kemacetan saat Jam Sibuk (Studi Kasus : Pasar Nou Kota Gunungsitoli dan Sekitarnya)," 2018.
[6] Andi Juansyah, "Pembangunan Aplikasi Child Tracker Berbasis Assisted – Global Positioning System (A-GPS) Dengan Platform Android," *J. Ilm. Komput. dan Inform.*, vol. 1, no. 1, pp. 1–8, 2015.
[7] D. E. Palupi, M. A. Akbar, and A. H. Brata, "Pengembangan Aplikasi Traffic Light E-Tilang Menggunakan Google Geofencing API Berbasis Android," *J. Pengemb. Teknol. Inf. dan Ilmu Komput. Univ. Brawijaya*, vol. 2, no. 2, pp. 6982–6988, 2018.
[8] M. S. Amri, "Membangun Sistem Navigasi Di Surabaya Menggunakan Google Maps Api," *Pens Its*, vol. 1, no. Proposal 2013, pp. 1–5, 2010.
[9] F. Masykur, "Implementasi Sistem Informasi Geografis Menggunakan Google Maps Api Dalam Pemetaan Asal Mahasiswa," *J. SIMETRIS*, vol. 5, no. 2, pp. 181–186, 2014.
[10] J. O. Agung, T. Efendi, and H. Agung, "Analisis Perbandingan Algoritma Floyd-Warshall Dengan Algoritma Bellman-Ford Dalam Pencarian Rute Terpendek Menuju Museum di Jakarta," *J. Sains dan Teknol.*, vol. 5, no. 1, pp. 1–7, 2018.