Buletin Sistem Informasi dan Teknologi Islam
Vol 2, No 3, Agustus 2021, pp. 197-207
ISSN: 2721-0901

Penerapan Metode *Dijkstra* Untuk Menentukan Lokasi Dan Jarak Tempuh Terpendek Kampus UMI Makassar

Suciwianingrat.H[a,1*], Yulita Salim[a,2], Nia Kurniati[a,3]

[a]*Program Studi Teknik informatika, Fakultas Ilmu Komputer, Universitas Muslim Indonesia, Jl.Urip Sumoharjo KM.05, Makassar dan 90231 ,Indonesia*
[1]*suciwianigrat98@gmail.com;* [2]*yulita.salim@umi.ac.id;* [3]*nia.kurniati@umi.ac.id;*
**corresponding author*

INFORMASI ARTIKEL

Diterima : 16 – 06 – 2021
Direvisi : 28 – 07 – 2021
Diterbitkan : 30 – 08 – 2021

Kata Kunci:
Aplikasi Pencarian Lokasi
Algoritma Dijkstra
Universitas Muslim Indonesia
Penentuan Jalur Terpendek

ABSTRAK

Mahasiswa seringkali kesulitan dalam menentukan jarak terpendek menuju ke Kampus UMI dikarenakan banyaknya jalur dapat dilalui oleh pengguna, maka dalam membantu menentukan jarak terpendek dapat digunakan peta konvensional dan memilih jalur yang dianggap terpendek dari daerah asal ke lokasi tujuan. Namun hal ini dirasa kurang maksimal dan memperlambat waktu karena harus memilih sendiri dari banyak jalur yang ada dan melakukan perhitungan sendiri jarak terpendek dari daerah asal menuju daerah tujuan yang dikehendaki. Algoritma *Dijkstra* merupakan salah satu algoritma untuk menentukan rute terpendek dari lokasi pengguna menuju lokasi tujuan dengan memasukkan variabel-variabel yang dibutuhkan yaitu *Latitude* dan *Longitude* untuk membuat titik simpul serta variabel jarak untuk menampilkan jarak titik antar simpul yang dibentuk dengan menerapkan metode *Haversine Formula.* Tujuan dari penelitian ini adalah menerapkan metode *dijkstra* dalam menentukan lokasi dan jarak tempuh terpendek ke kampus UMI di Kota Makassar berbasis *android*. Proses pencarian rute akan menggunakan algoritma *dijkstra* dan hasil akhir yang ditampilkan adalah informasi rute jalan yang harus dilalui. Hasil dari penelitian ini adalah berdasarkan hasil kuesioner yang di berikan ke mahasiswa didapatkan tingkat persentase sebanyak 85,2% yang setuju dengan aplikasi tersebut, serta akurasi penerapan metode *dijkstra* dengan tingkat persentase 85,74 % pada sistem yang telah dibangun.

I. Pendahuluan

Mahasiswa seringkali kesulitan menentukan jarak terpendek menuju ke kampus UMI dikarenakan banyaknya jalur dapat dilalui oleh pengguna, maka dalam membantu menentukan jarak terpendek dapat digunakan peta konvensional dan memilih mana jalur yang dianggap terpendek dari daerah asal ke daerah tujuan. Namun hal ini dirasa kurang maksimal dan memperlambat waktu karena harus memilih sendiri dari banyak jalur yang ada dan melakukan perhitungan sendiri mana kira-kira jarak terpendek dari daerah asal menuju daerah tujuan yang dikehendaki.

Kemajuan teknologi saat ini tentunya sangat memudahkan masyarakat untuk mencari informasi, salah satunya yaitu pencarian informasi terkait perguruan tinggi yang diminati baik itu fasilitas, alamat, dan jalan menuju lokasi. Namun pada kenyataannya masih banyak dari masyarakat baik berasal dari dalam maupun luar kota Jakarta yang belum mengetahui lokasi dan rute yang harus dilewati untuk menuju ke perguruan tinggi dengan program Studi Teknik Informatika di Jakarta [1].

Oleh karena itu, penulis tertarik melakukan penelitian terkait rute terpendek menggunakan Algoritma *Dijkstra*. Algoritma ini dipilih karena dapat menyelesaikan pencarian jalur terpendek dari satu simpul ke semua simpul yang ada pada suatu graf berarah dengan bobot dan nilai tidak negative[2]. Algoritma *Dijkstra* lebih *intensif* dalam komputasi untuk pencarian jalur optimum dalam suatu jaringan seperti internet, dan waktu rata-rata eksekusi algoritma *Dijkstra* lebih kecil dibandingkan algoritma *Ant Colony*, maka algoritma *Dijkstra* banyak digunakan dalam pencarian jalur optimum pada jaringan internet dibanding algoritma lain[3].

II. Metode

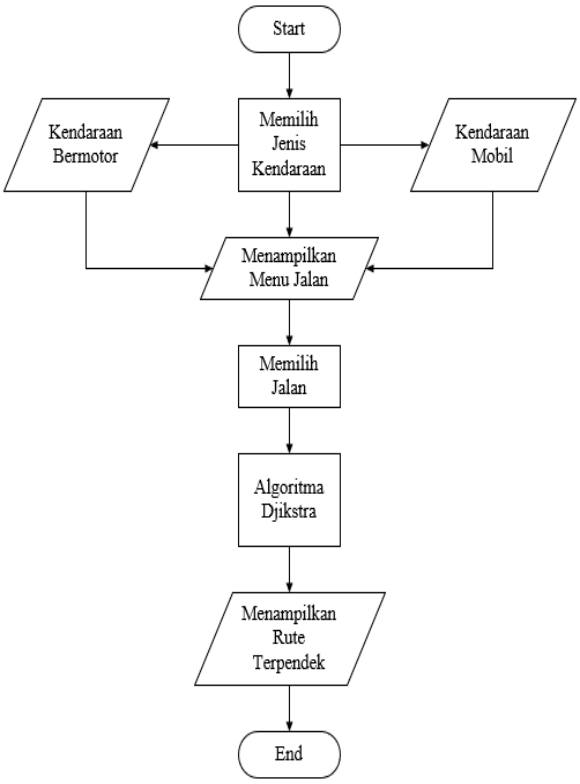
Algoritma *Dijkstra* salah satu algoritma yang efektif dalam memberikan lintasan terpendek dari suatu lokasi ke lokasi yang lainnya [4]. Prinsip dari algoritma *Dijkstra* adalah mencari titik lokasi dengan pencarian dua lintasan yang paling pendek. Pada setiap iterasi, jarak titik yang diketahui (dari titik awal) diperbaharui bila

ternyata didapat titik yang baru yang memberikan jarak terpendek. Syarat algoritma ini adalah bobot sisinya yang harus non-negatif [5].

A. Langkah-langkah Metode *Dijkstra* [4]–[6]

1. 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
2. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi).
3. Set semua node yang belum dilalui dan set node awal sebagai "Node keberangkatan".
4. Dari node keberangatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru..
5. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai "Node dilewati". Node dilewati". Node yang dilewati tidak akan pernah dicek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
6. Set "Node belum dilewati" dengan jarak terkecil (dari node keberangkatan) sebagai "Node keberangkatan " selanjutnya dan ulangi langkah 5.

B. Analisis Sistem yang diusulkan



Gambar 1 Sistem yang diusulkan

Pada Gambar 1. diatas sistem yang akan diusulkan user memilih jenis kendaraan yang akan dipakai baik itu kendaraan bermotor maupun mobil setelah itu akan menampilkan menu jalan yang terdiri dari beberapa nama jalan ketika user memilih jalan tersebut maka akan menampilkan rute menuju ke lokasi Kampus UMI yang menerapkan metode *dijkstra* dalam mencari rute terpendek.

III. Hasil dan Pembahasan

A. Pengujian Akurasi Metode

Tabel 1 Uji Coba Aplikasi

No	Nama Jalan	Jarak pada sistem	Jarak sebenarnya	Akurasi
1.	Jln A.P Pettarani II	1 Km	2.5 Km	40%
2.	Jln Sukaria 11	861 M	950 M	90,63%
3.	Jln Abdullah Daeng Sirua	3 Km	3.0 Km	100%
4.	Jln Batua Raya	3.1 Km	3.2 Km	93,75%
5.	Jln A.P Pettarani No 9	3 Km	4.9 Km	61,22%
6.	Jln Perintis Kemerdekaan IV	5.5 Km	5.6 Km	89,28%
7.	Jln BTN Bung Permai	5.8 Km	5.4 Km	92,59%
8.	Jln Manggala Raya no 256	8 Km	8 Km	100%
9.	Jln Dr. Ratulangi	7 Km	7 Km	100%
10.	Jln Skarda N Lorong 1	7 Km	7 Km	100%

Diatas penulis akan mencari tingkat akurasi dari masing masing nama jalan dengan membandingkan jarak yang ada pada sistem dengan jarak sebenarnya yang ada di *Google Maps*, Untuk mengetahui tingkat keakuratan sistem maka dilakukan penjumlahan jarak pada sistem dibagi jarak sebenarnya dikali 100% kemudian hasil persentasenya dijumlahkan kemudian dibagi dengan jumlah jalan.

$$\frac{40 + 90{,}6315 + 100 + 93{,}75 + 61{,}22 + 89.28 + 92{,}59 + 100 + 100 + 85{,}71}{10}$$

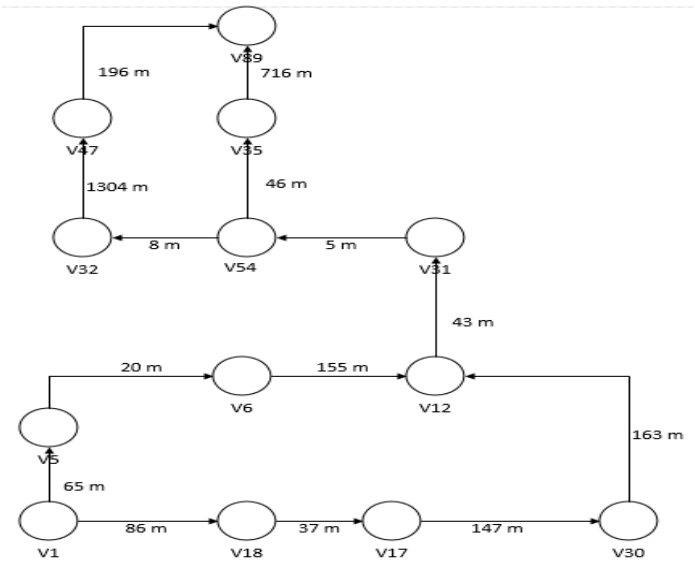
$$= \frac{867{,}47}{10} = 85{,}74\ \%$$

B. Penerapan Metode *Dijkstra*

Tabel 2 Pengujian Graph Jalan

No	Titik	Nama Lokasi	Long	Lat
1	V1	Pondok Massituru	-5.147	119.44592
2	V5	Kedai Kopi Sengkaya	-5.14658	119.44599
3	V6	Cargonia Express	-5.1465	119..44615
4	V12	Perempatan Jln Racing Centre 2 dan Jln M.Saleh Yusuf	-5.14588	119.44739
5	V17	Tusukkan Indonesia	-5.14711	119.44675
6	V18	Berkah Racing	-5.1474	119.44658
7	V30	Jln Inspeksi Kanal	-5.14725	119.44791
8	V31	Jln Racing Centre 2	-5.14581	119.44777
9	V32	Pengobatan Bugis Kunfayakun	-5.1458	119.44788
10	V35	Jln Racing Centre	-5.14608	119.45019
11	V47	Jalan Urip Sumoharjo	-5.14028	119.4505
12	V54	Jln Racing Sinrijala	-5.1458	119.44781
13	V89	Universitas Muslim Indonesia	-5.13976	119.44881

Berdasarkan Tabel 2 diatas menunjukkan nama jalan yang dilewati menuju Kampus UMI, nama jalan tersebut berdasarkan dari titik koordinat latitude dan longitude yang telah dibuat sebelumnya dan dari titik koordinat tersebut akan membentuk sebuah graph seperti pada Gambar dibawah ini.



Gambar 2 Graf Jl A.P Pettarani II

Pada Gambar 2 diatas dapat dilihat rute-rute menuju Kampus UMI yang terdapat tiga jalur alternatif yaitu:

Tabel 3 Jalur Andi Pettarani II

No	Rute	Jalur
1	Jalur 1	V1 - V18 - V17 - V30 - V12 – V31- V54 – V35 – V89 = 1.243 m
2	Jalur 2	V1 - V5 - V6 - V12 - V31 - V54 - V35 - V89 = 1.050 m
3	Jalur 3	V1 - V5 - V6 - V12 - V31 - V54 - V32 - V47 – V89 = 1.796 m

Pada Tabel 3 diatas menunjukkan terdapat tiga jalur untuk menuju Kampus UMI ketika pengguna berada di Jalan Andi Pettarani II dan dari setiap jalur tersebut akan melewati masing-masing titik simpul dimana titik simpul tersebut berdasarkan pada Tabel 2 sebelumnya.

Berdasarkan rute diatas maka jalur terpendek dari Jalan Andi Pettarani II menuju Kampus UMI terletak pada jalur 2 yaitu melewati rute V1-V5-V6-V12-V31-V54-V35-V89 dengan jarak 1.050 meter .

Tabel 4 Hasil Perhitungan Algortima *Dijkstra*

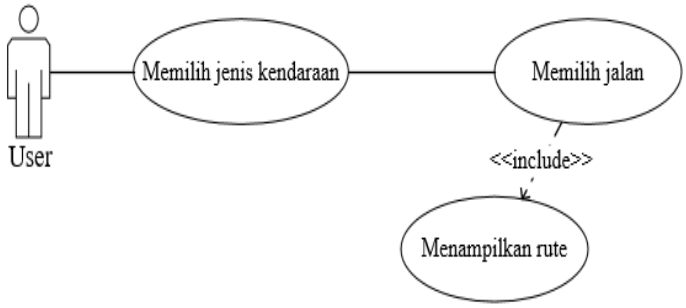
No	Sumber Simpul	Rute Simpul	Jarak antar Simpul
1	V1	V1-V5	65 m
		V1- V18	86 m
2	V5	V5-V6	20 m + 65 m = 85 m
3	V18	V18-V17	86 m + 37 m = 123 m
4	V6	V6-V12	85 m + 155 m = 240 m
5	V17	V17-V30	123 m + 147 m = 270 m
6	V30	V30-V12	270 m + 163 m = 433 m
7	V12	V12 – V31	240 m + 43 m = 283 m
8	V31	V31-V54	283 m + 5 m = 288 m
9	V54	V54-V32	288 m + 8 m = 296 m
		V54–V35	288 m + 46 m = 334 m
10	V32	V32-V47	296 m + 1304 m = 1600 m
11	V35	V35-V89	334 m + 716 m = 1050 m
12	V47	V47- V89	1600 m + 196 m = 1796 m

Hasil perhitungan dari algoritma *dijkstra* telah diimplementasikan kedalam sistem pada penelitian ini. Hasil yang ditampilkan sistem dengan contoh kasus ini telah sesuai dengan perhitungan manual algoritma *dijkstra.* Seperti pada tabel diatas.

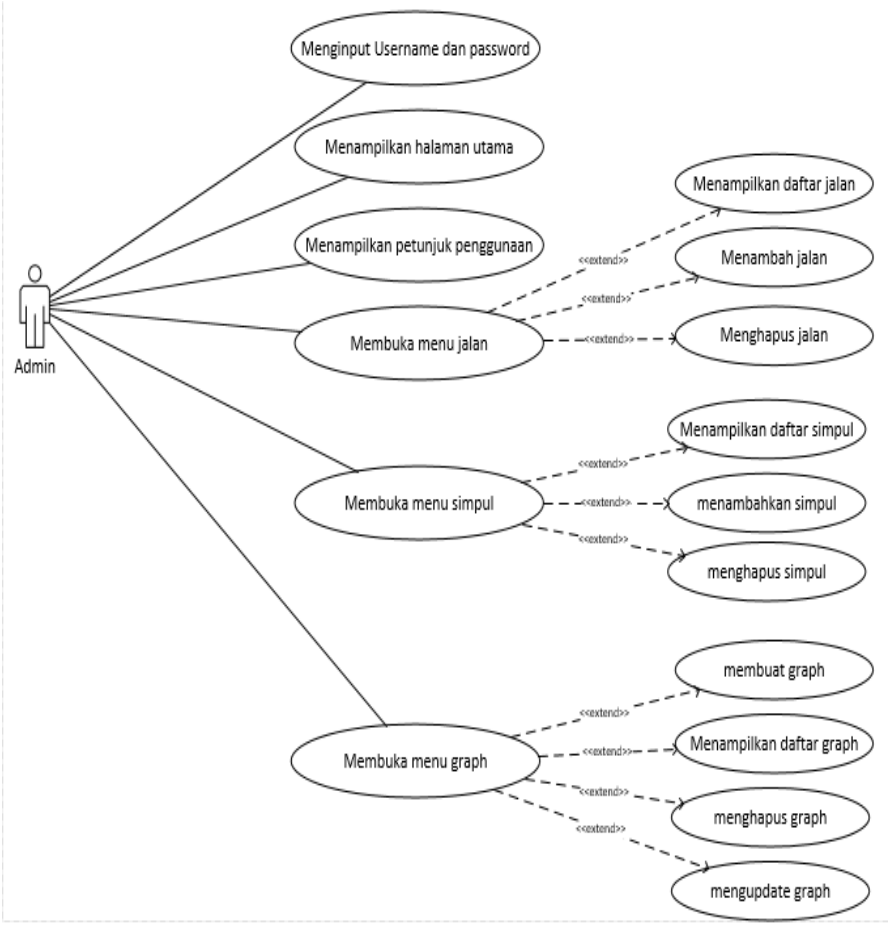
C. Perancangan Sistem

Perancangan sistem yang dilakukan yaitu membuat perancangan *Unifed Modeling Language* (UML) dengan menggunakan beberapa diagram, yaitu *Use* Cace *Diagram*, *Activity Diagram Sequence Diagram dan Class diagram.*

1. *Use* Cace *Diagram*
   Use Case Diagram digunakan untuk mengetahui fungsi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Adapun *use case diagram* dari sistem yang diusulkan sebagai berikut :



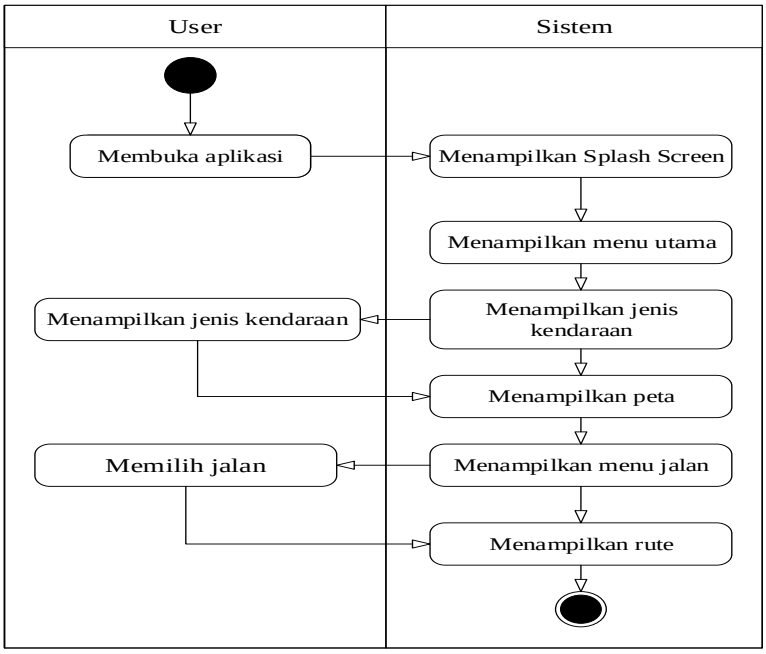
Gambar 3 *Use* Cace *Diagram* (*User*)



Gambar 4 *Use* Cace *Diagram* (*Admin*)

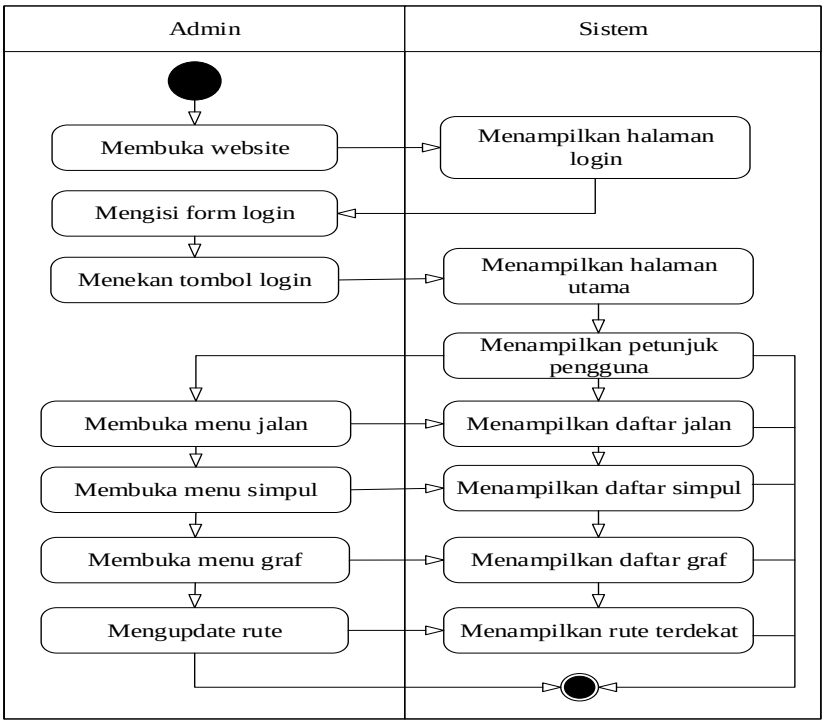
2. Activity *Diagram*

Activity Diagram untuk mendefinisikan alur kerja kapan dimulai aktivitas yang terjadi selama proses sistem yang berjalan dan urutan kejadian. Adapun *activity diagram* dari sistem yang diusulkan sebagai berikut :



Gambar 5 *Activity Diagram (User)*

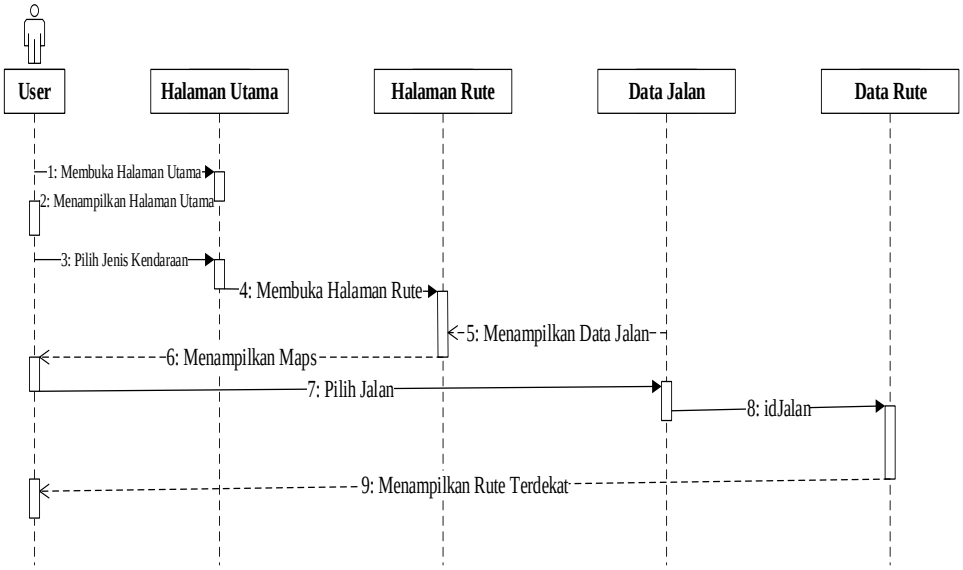
Pada Gambar 5 diatas mulanya *user* (masyarakat) membuka aplikasi dan sistem menampilkan menu utama, pada menu utama akan ditampilkan pilihan jenis kendaraan yaitu kendaraan bermotor maupun mobil kemudian pengguna memilih jenis kendaraan yang akan digunakan untuk menuju Kampus UMI selanjutnya pengguna memilih nama jalan dari lokasi mereka berada dan sistem akan menampilkan rute terpendek menuju Kampus UMI berdasarkan jenis kendaraan dan nama jalan yang dipilih.



Gambar 6 *Activity Diagram (Admin)*

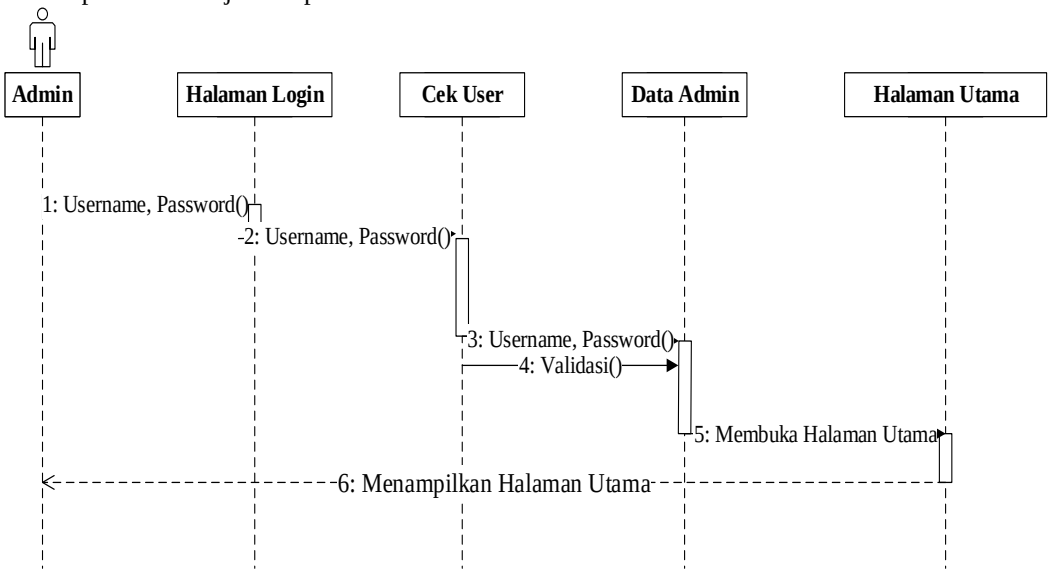
Pada Gambar 6 diatas *admin* membuka *website* dan akan sistem akan menampilkan halaman *login*, pada halaman tersebut *admin* mengisi form *login* dengan memasukkan username dan *password* jika berhasil *login* maka akan ditampilkan halaman utama dari website tersebut, pada halaman utama *admin* dapat mengelola data seperti nama jalan, titik simpul, dan graph yang saling berhubungan dengan aplikasi *android* pengguna dalam menentukan rute terpendek

3. *Sequence Diagram*



Gambar 7 *Sequence Diagram(User)*

Pada Gambar 7 diatas *user* membuka halaman utama dan akan di tampilkan pilihan jenis kendaraan selanjutnya user memilih jenis kendaraan yaitu kendaraan bermotor ataupun mobil kemudian dari jenis kendaraan yang dipilih akan di tampilkan halaman rute yang menampilkan nama-nama jalan, *user* kemudian memilih jalan yang sesuai dengan lokasi mereka dan sistem akan menampilkan rute terpendek menuju Kampus UMI

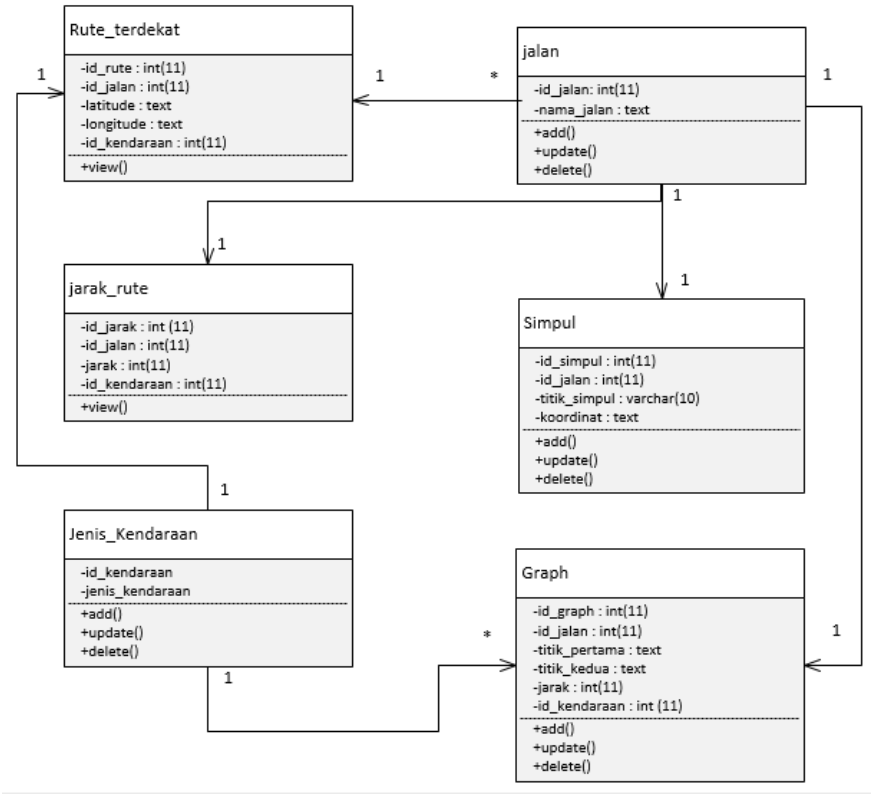


Gambar 8 *Sequence Diagram(Admin)*

Pada Gambar 8 diatas admin memasukkan username dan password untuk bisa masuk ke aplikasi ketika username dan *password* sudah sesuai maka akan masuk ke halaman utama aplikasi.

4. *Class Diagram*

Class diagram adalah model statis yang menggambarkan struktur dan deskripsi class serta hubungannya antara class. Class terdiri dari nama kelas, atribut dan operasi/methode. Berikut adalah class diagram sistem yang telah dibuat



Gambar 9 *Class Diagram* Aplikasi

Pada Gambar 9 diatas terdapat 6 *field* yang saling berhubungan, pada *field* jalan saling keterkaitan dengan titik simpul dikarenakan nama jalan akan terbentuk dari titik simpul yang saling berhubungan yang membentuk sebuah graph, kemudian pada *field* jenis kendaran berkaitan dengan rute terpendek serta jarak rute dimana sistem akan menampilkan rute terpendek dan jarak rute berdasarkan jenis kendaraan yang dipilih.

D. Implementasi *Interface* Sistem

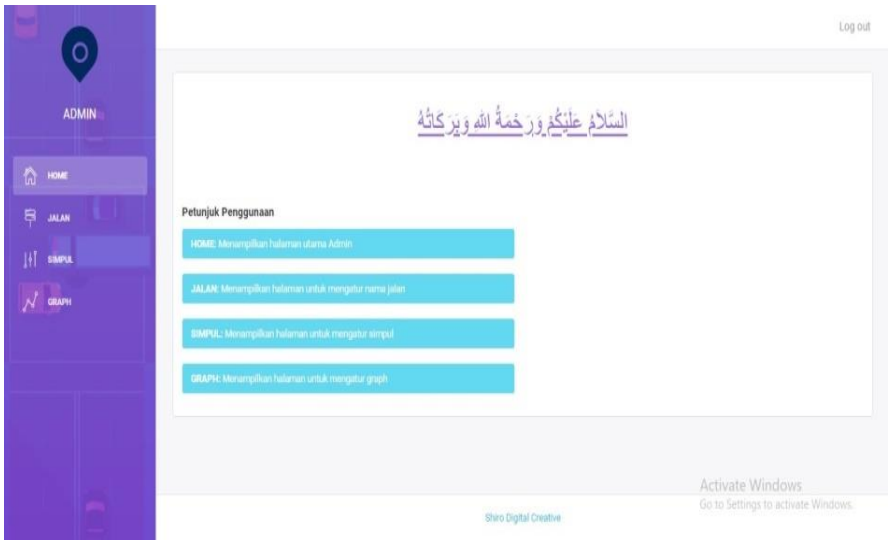
Implementasi Sistem adalah kelanjutan dari kegiatan perancangan sistem dan dapat dipandang sebagai usaha untuk mewujudkan sistem yang dirancang. Berikut ini merupakan tempilan-tampilan implementasi dari tahap perancangan.

1. Antarmuka Menu Maps
   Antarmuka Menu Maps merupakan Halaman yang menampilkan peta dan memilih lokasi.

Gambar 10 Antarmuka Menu Maps (*User*)

Pada Gambar 10, memperlihatkan tampilan Antarmuka Menu Maps dimana disini terdapat pilihan lokasi sebagai titik awal.

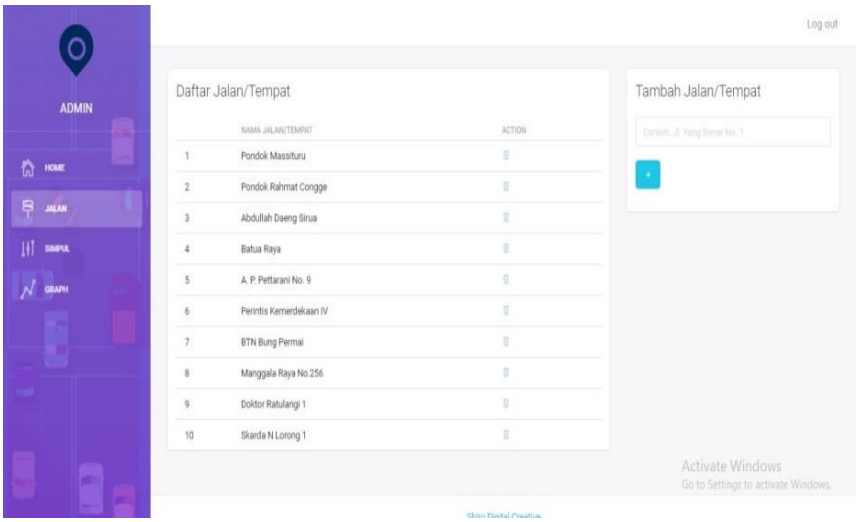
2. Antarmuka Halaman Menu Utama
Halaman Menu Utama akan tampil ketika *admin* berhasil login.



Gambar 11 Antarmuka Menu Utama(*Admin*)

Pada Gambar 11, memperlihatkan tampilan Antarmuka Halaman Menu Utama. Pada halaman menampilkan petunjuk pengguna yang terdiri dari Home, Jalan, Simpul, dan Graph.

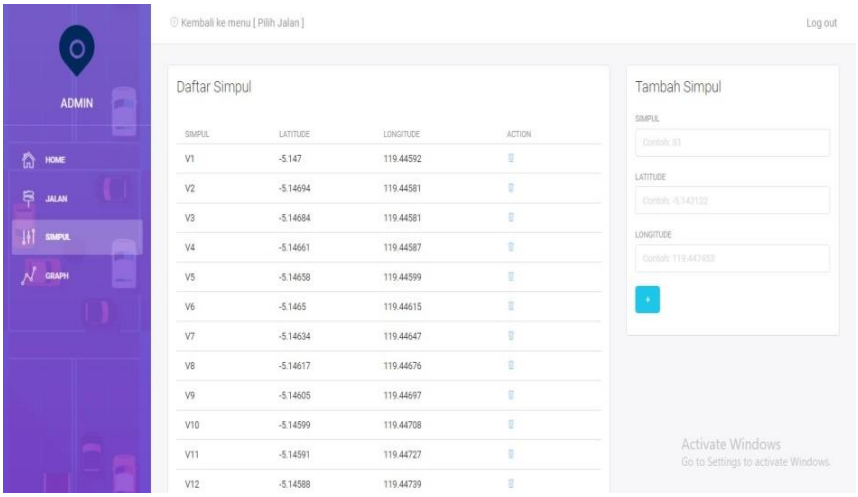
3. Antarmuka Halaman Menu Jalan
Antarmuka Halaman Menu Jalan akan ditampilkan ketika admin memilih menu Jalan.



Gambar 21 Antarmuka Menu Jalan(*Admin*)

Pada Gambar 21, memperlihatkan tampilan Antarmuka Halaman Menu Jalan.

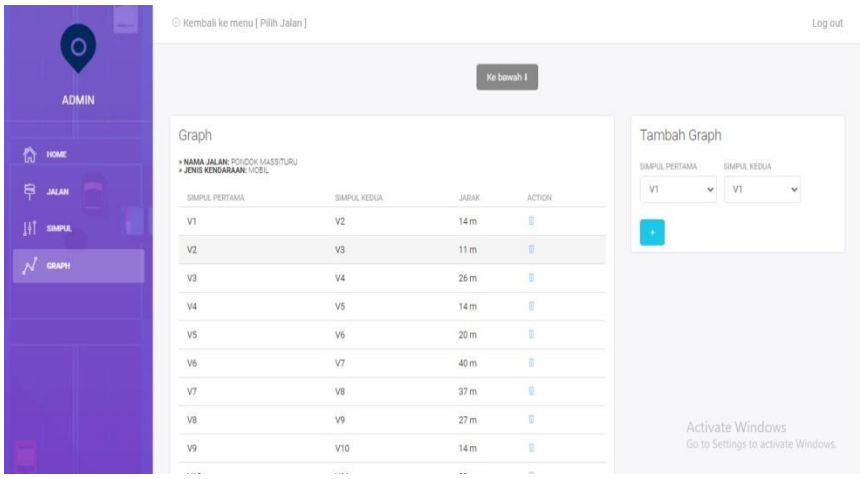
4. Antarmuka Halaman Menu Simpul
Antarmuka Halaman Menu Simpul akan ditampilkan ketika admin memilih menu Simpul



Gambar 22 Antarmuka Menu Simpul(*Admin*)

Pada Gambar 22, memperlihatkan tampilan Antarmuka Halaman Menu Simpul. Pada halaman menampilkan titik koordinat dari titik jalan yang sudah ditentukan.

5. Antarmuka Halaman Menu Graph
Antarmuka Halaman Menu Graph akan ditampilkan ketika admin memilih menu Graph.



Gambar 23 Antarmuka Menu Graph(*Admin*)

Pada Gambar 4.5, memperlihatkan tampilan Antarmuka Halaman Menu Graph. Pada halaman ini menampilkan jarak antar simpul yang saling berhubungan yang dalam menentukannya menggunakan metode *haversine.*

IV. Kesimpulan Dan Saran

Setelah melakukan analisis dan perancangan sistem maka penulis dapat mengambil kesimpulan bahwa aplikasi ini dapat menampilkan informasi tentang pencarian lokasi dan jarak terdekat dengan menerapkan metode *Dijkstra.* Titik lokasi yang dapat terlihat dalam sistem adalah Jl. A.P. P ettarani II, Jl.Sukaria 11, Jl. Abdullah Daeng Sirua, Jl. Batua Raya, Jl. A.P. Pettarani No.9, Jl. Perintis Kemerdekaan IV, Jl. BTN Bung Permai, Jl. Manggala Raya No.256, Jl. Doktor Ratulangi 1, Jl. Skarda N Lorong 1. Aplikasi dapat berjalan menggunakan perangkat mobile yang menggunakan sistem operasi *Android* 5.0 Lollipop Kuesioner menunjukkan bahwa sebesar 85,2 % mahasiswa setuju dengan aplikasi yang sudah dibuat untuk menentukan jarak terdekat menuju Kampus UMI yang melibatkan 10 orang responden. Berdasarkan pengujian yang telah dilakukan persentase akurasi metode dengan jarak sebenarnya memiliki tingkat akurasi 85.74% yang berarti jarak hasil perhitungan sistem hampir mendekati jarak aslinya. Sebaiknya kedepan dapat menggunakan Algoritma ACO untuk melihat penggunaan waktu yang cepat menuju rute tujuan.

Daftar Pustaka

[1] Sauwani, A. Halim, Jeki, and V. N. Putra, "Implementasi Algoritma Djikstra Untuk Menentukan Lokasi Dan Jarak Tempuh Terpendek Kampus It Di Jakarta," *J. Inform.*, vol. 6, no. 1, pp. 29–36, 2019.

[2] Bogas and Shagas, "Algoritma Dijkstra untuk Penentuan Jalur Terdekat dan Rekomendasi Obyek Parawisata di Pulau Bali," no. Universitas Dian Nuswantoro : Program Studi Teknik Informatika, 2008.

[3] Fitria, "Implementasi Algoritma Dijkstra dalam Aplikasi untuk Menentukan Lintasan Terpendek Jalan Darat Antar Kota di Sumatera Bagian Selatan," *J. Sist. Inf.*, vol. 5, no. 2, 2013.

[4] Pranatawijaya, V. Handrianus, W. Widiatry, N. N. K. Sari, and P. B. A. A. Putra, "Sistem Informasi Geografis Mencari Rute Lokasi Travel Di Kota Palangka Raya Berbasis Website," *J. Teknol. Inf. J. Keilmuan dan Apl. Bid. Tek. Inform.*, vol. 13, no. 1, pp. 76-82., 2019.

[5] Pinandita, Tito, and L. N. Arifin, "Sistem Informasi Geografis Pencucian Sepeda Motor Melalui Algoritma Dijkstra Berbasis Android Di Kota Purwokerto," *J. Media Pratama*, vol. 14., no. 1, pp. 53–61, 2020.

[6] Hamdi, Saeful, and P. Prihandoko, "Analisis Algoritma Dijktra dan Algoritma Bellman-Ford Sebagai Penentuan Jalur Terpendek Menuju Lokasi Kebakaran (Studi Kasus: Kecamatan Praya Kota)," *Energy*, vol. 8, no. 1, pp. 26-32., 2018.