RANCANG BANGUN APLIKASI PENCARIAN RUTE TERPENDEK TAMBAL BAN MENGGUNAKAN ALGORITMA DIJKSTRA

Daniel Nusalawo[1], Angelia M. Adrian*[1], Junaidy B. Sanger[1]
[1]Program Studi Teknik Informatika; Fakultas Teknik
[1]Universitas Katolik De La Salle Manado
e-mail: 14013022@unikadelasalle.ac.id, *madrian@unikadelasalle.ac.id, jsanger@unikadelasalle.ac.id

***Abstrak*— Informasi lokasi tambal ban merupakan informasi yang sangat penting bagi pengendara kendaraan bermotor. Bagi pengendara yang kurang mengetahui daerah sekitar ketika mengalami kebocoran ban, maka tentunya akan menyulitkan untuk mencari tempat tambal ban terdekat. Penelitian ini bertujuan untuk mengimplementasikan algoritma pencarian rute terpendek (Algoritma Dijsktra) pada kasus tambal ban di Kota Manado melalui sebuah aplikasi sehingga dapat membantu para pengendara menemukan rute terpendek menuju lokasi tambal ban.**

***Kata Kunci*—Rute Terpendek, Algoritma Dijkstra, Tambal Ban, Aplikasi**

I. PENDAHULUAN

Seiring dengan penggunaan kendaraan bermotor yang terus bertambah, pilihan transportasi umum juga terus berkembang dan tumbuh pesat, akan tetapi perkembangan tersebut tidak diimbangi dengan pembangunan infrastruktur jalan yang memadai oleh pemerintah. Salah satunya di Kota Manado, Ibu Kota dari Provinsi Sulawesi Utara. Penggunaan kendaraan bermotor di jalanan yang meningkat perlu diperhatikan pula kondisi kendaraan saat berkendara. Pengecekan kondisi kendaraan sebelum digunakan itu merupakan hal yang sangat penting. Hal ini dilakukan untuk menjamin keselamatan dan kenyamanan berkendara, namun seringkali hal ini sering terlewatkan. Biasanya masalah yang paling sering ditemukan saat berkendara adalah ban kempis atau ban bocor. Penyebabnya antara lain karena menurunnya tekanan udara pada ban, atau faktor jalanan yang rusak dan terdapat material tajam seperti paku.

Tambal ban merupakan tempat yang penting jika kendaraan bermotor seseorang mengalami kebocoran ban atau ban kempis. Keberadaan tempat tambal ban begitu sangat penting bagi seseorang untuk menambal ban kendaraan atau untuk menambah tekanan udara pada ban. Pada waktu mengalami kebocoran ban atau kempis, pengendara ingin mencari tempat tambal ban yang terdekat dari lokasi untuk dikunjunginya. Namun, permasalahan mengenai ketidaktahuan lokasi-lokasi tambal ban membuat pengendara kesulitan dalam menentukan rute yang harus ditempuh agar bisa sampai ke lokasi tambal ban terdekat. Kejadian ini akan sangat menyulitkan bagi pengendara yang masih belum mengetahui wilayah sekitar khususnya di Manado.

Pencarian rute terpendek ke lokasi tambal ban, pengendara hanya perlu memilih titik lokasi awal dan titik lokasi tujuan sehingga bisa mencari lokasi tujuan dalam waktu yang cepat. Untuk mencari rute terpendek ini dibutuhkan sebuah algoritma yang dapat mempermudah untuk memilih jalur yang akan dilalui. Salah satu algoritma untuk mencari rute terpendek adalah algoritma Dijkstra. Terdapat penelitian [1] untuk penentuan lintasan terpendek ke Kota Klaten dengan membandingkan algoritma Dijkstra dan Algoritma Warshal. Dari hasil penelitian tersebut disimpulkan bahwa algoritma Dijkstra lebih unggul karena waktu pemrosesan yang lebih cepat dan lebih dekat dengan jarak sebenarnya. [2] telah melakukan penelitian dengan membuat suatu aplikasi pencarian rute terpendek yang mengimplementasikan Algoritma Dijkstra untuk pencarian tambal ban di Kota Medan. Aplikasi yang dibuat berbasis Android. [3] telah melakukan penerapan Algoritma Dijkstra dalam sebuah aplikasi untuk mencari rute terpendek jalan darat antar kota di pulau Sumatera bagian selatan. Hasil dari penelitian tersebut dapat menawarkan beberapa kemudahan dalam menyusun peta secara dinamik.

Adapun tujuan dilakukannya penelitian ini adalah mengimplementasikan algoritma pencarian rute terpendek (algoritma dijsktra) pada kasus tambal ban di Kota Manado melalui sebuah aplikasi sehingga dapat membantu para pengendara menemukan rute terpendek menuju lokasi tambal ban.

II. TINJAUAN PUSTAKA

Rute Terpendek

Proses meminimalkan bobot di suatu rute graf adalah arti dari kata terpendek pada persoalan suatu lintasan. Beberapa tipe masalah rute terpendek sebagai berikut:

a. Rute terpendek antara dua buah node.
b. Rute terpendek antara seluruh pasangan node.
c. Rute terpendek dari node terpilih melewati seluruh node lainnya.
d. Rute terpendek antara dua buah node yang melewati sebagian node terpilih.

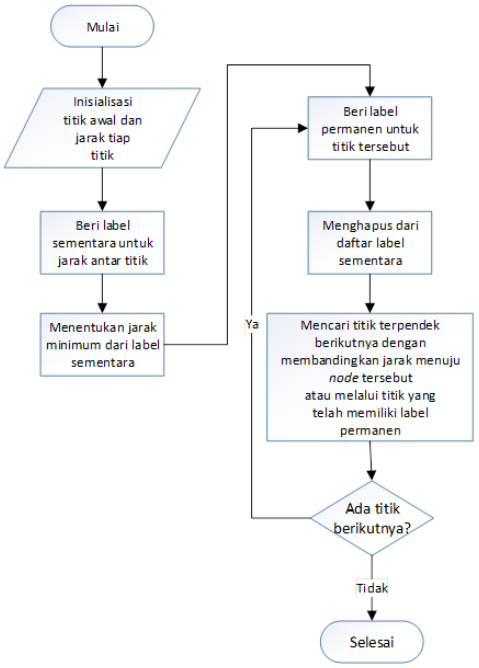
Suatu graf dideskripsikan untuk himpunan simpul dan himpunan sisi. Simpul menyatakan entitas-entitas data dan sisi menyatakan keterhubungan antar simpul [4]. Untuk graf G dipakai notasi matematika dengan Persamaan (1).

$$G = (V,E) \quad (1)$$

Keterangan :
G = Graf
V = Verteks atau Simpul atau Titik atau *Node*
E = *Edge* atau *Arc* atau Busur

Algoritma Dijkstra

Penggunaan dari Algoritma ini adalah untuk menghitung rute terdekat dari node awal menuju node tujuan dengan mengetahui jarak rute gabungan setiap titik. Proses dari Algoritma Dijkstra yang berbentuk Flowchart ditunjukan pada Gambar 1.



Gambar. 1. *Flowchart* Algoritma Dijkstra

Tahapan penyelesaian pada algoritma Dijkstra adalah sebagai berikut: Misalnya sebuah graf berbobot dengan *n* buah verteks dinyatakan dengan matriks ketetanggaan $M = [m_{ij}]$, yang dalam hal ini,

m_{ij} = bobot sisi (*i*,*j*) (pada graf tak-berarah $m_{ij} = m_{ji}$)
$m_{ii} = 0$
$m_{ij} = \infty$, jika tidak ada sisi dari verteks *i* ke verteks *j*

Selain matriks *M*, juga memakai tabel $S = [s_i]$ yang dalam perihal ini,

$s_i = 1$, apabila verteks *i* termasuk ke dalam jalur terdekat
$s_i = 0$, apabila verteks *i* tidak termasuk ke dalam jalur terdekat dan tabel $D = [d_i]$ yang dalam perihal ini,
d_i = panjang jalur dari verteks awal *a* menuju verteks *i*

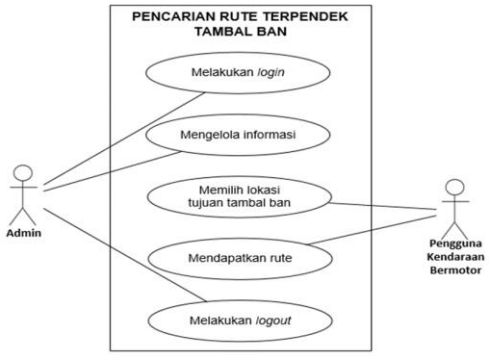
III. METODE PENELITIAN

Adapun Metodologi yang digunakan adalah metodologi Waterfall dengan kakas pemodelan *Unified Modelling Language* (UML) berupa *Use Case Diagram*, *Class Diagram*, dan *Activity Diagram*. *Waterfall* adalah metodologi pengembangan perangkat lunak yang digunakan. *Waterfall* sering juga disebut model *sequential linear* atau *classic life cycle*. Bentuk *waterfall* menyediakan pendekatan siklus hidup *software* secara berurutan dimulai dari analisis, desain, pengodean, dan pengujian adalah sebagai berikut [5]:

1. Tahap analisis merupakan tahap identifikasi layanan, batasan dan pengumpulan data dari berbagai studi pustaka, guna mengetahui keperluan akan sistem yang bakal dibangun demi menyelesaikan permasalahan yang selama ini.
2. Tahap desain merupakan tahap memodelkan rancangan sistem berdasarkan hasil analisis dari tahap sebelumnya. Pada tahap ini memodelkan rancangan sistem baru yang berbentuk aplikasi.
3. Tahap pengodean merupakan tahap implementasi dari tahapan desain. Tahap ini merupakan tahapan pembangunan program, *coding*, integrasi dan *system testing*.
4. Tahap pengujian merupakan tahapan testing sistem yang baru berbentuk aplikasi. Pelatihan berupa pengenalan fitur-fitur yang disediakan oleh aplikasi berdasarkan jenis-jenis pengguna.

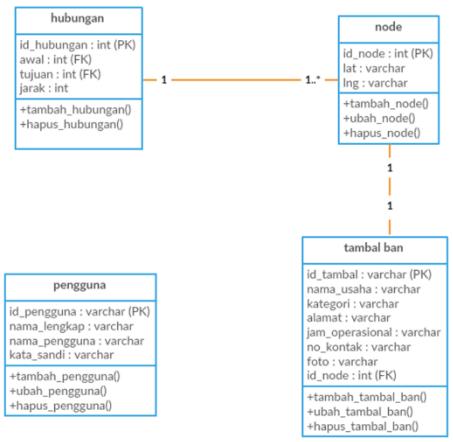
IV. HASIL DAN PEMBAHASAN

Berdasarkan hasil analisa, maka dapat dimodelkan sistem baru yang dibangun dengan menggunakan *Use Case Diagram*, *Class Diagram*, dan *Activity Diagram*. *Use Case Diagram* menggambarkan hubungan di antara pengguna dan sistem yang dibangun. *Use Case Diagram* sistem baru dapat dilihat pada Gambar 2. Terdapat 2 level pengguna yaitu Admin dan Pengguna Kendaraan Bermotor.



Gambar. 2. *Use Case Diagram* Sistem Baru

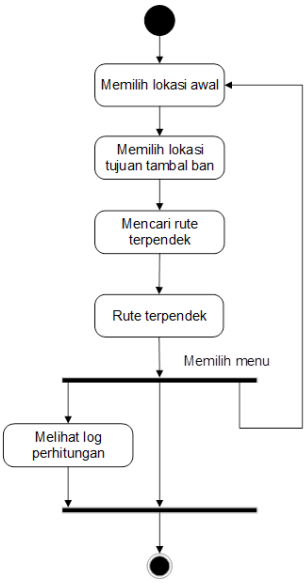
Identifikasi interaksi antar objek dan atribut serta mengklasifikasikannya demi mempermudah pembuatan basis data pada tahap implementasi dan itu dapat terlihat pada perancangan *Class Diagram* yang ditunjukan pada Gambar 3.



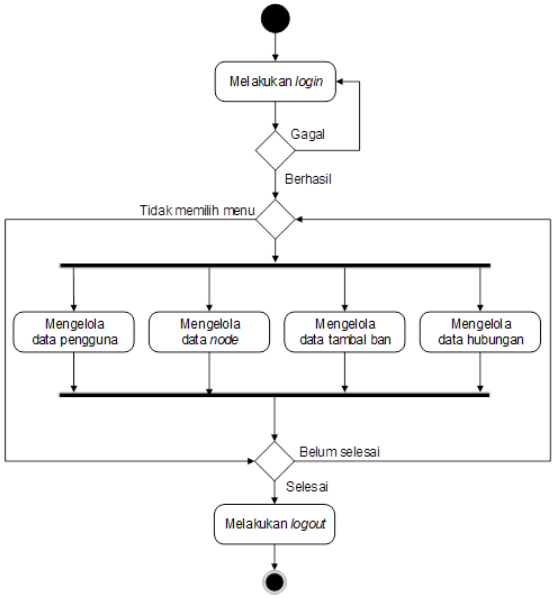
Gambar. 3. *Class Diagram* Sistem Baru

Activity Diagram digunakan untuk menggambarkan tentang aliran kerja dan aktivitas dari sistem baru yang akan dibangun.

Activity Diagram Sistem Baru untuk Pengguna Kendaraan Bermotor dan Admin dapat dilihat pada Gambar 4 dan Gambar 5.



Gambar. 4. *Activity Diagram* Sistem Baru (Pengguna Kendaraan Bermotor)



Gambar. 5. *Activity Diagram* Sistem Baru (Admin)

Perancangan Antarmuka

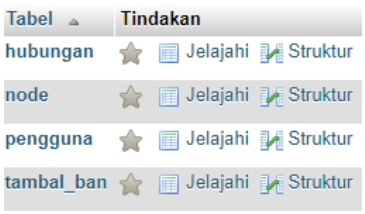
Tahap ini menjelaskan mengenai rincian dari rancangan berupa *storyboard* untuk beberapa halaman pada sistem yang akan dibangun. Halaman-halaman yang dimaksud antara lain:

1. *Storyboard* halaman beranda
   Halaman beranda ini merupakan tampilan yang muncul saat pengguna mengakses aplikasi. Halaman ini juga berfungsi sebagai tempat untuk mencari rute terpendek menuju lokasi tambal ban yang dipilih pengguna.
2. *Storyboard* halaman *node*
   Storyboard halaman *node* adalah tampilan saat admin ingin menambah, mengubah atau menghapus data *node*. Atau dengan kata lain, halaman ini berfungsi sebagai tempat untuk mengelola data *node* pada aplikasi.
3. *Storyboard* halaman tambah *node*
   Storyboard halaman tambah *node* adaalah tampilan saat admin menekan tombol tambah *node*. Halaman ini berfungsi sebagai tempat untuk menambah *node* baru pada aplikasi.
4. *Storyboard* halaman tambal ban
   Storyboard halaman tambal ban adalah tampilan saat admin ingin menambah, mengubah atau menghapus data tambal ban. Halaman ini berfungsi sebagai tempat untuk mengelola data tambal ban pada aplikasi.
5. *Storyboard* halaman tambah tambal ban
   Storyboard halaman tambah tambal ban adalah tampilan saat admin menekan tombol tambah tambal ban. Halaman ini berfungsi sebagai tempat untuk menambah tambal ban baru pada aplikasi.
6. *Storyboard* halaman hubungan
   Storyboard halaman hubungan adalah tampilan saat admin ingin menambah atau menghapus data hubungan. Halaman ini berfungsi sebagai tempat untuk mengelola data hubungan pada aplikasi.
7. *Storyboard* halaman tambah hubungan
   Storyboard halaman tambah hubungan adalah tampilan saat admin menekan tombol tambah hubungan. Halaman ini berfungsi sebagai tempat untuk menambah hubungan antar *node* yang baru pada aplikasi.

Implementasi

Tahap ini merupakan implementasi dari perancangan yang telah dibuat sebelumnya yaitu implementasi basis data, pemrograman dan implementasi antarmuka. Tahap pengodean adalah tahap implementasi dari tahapan desain yang merupakan tahapan pembangunan program, *coding*, integrasi dan *system testing*.

Struktur basis data yang dibuat menggunakan phpMyAdmin dan implementasinya dapat dilihat pada Gambar 6 yang menunjukkan daftar tabel pada basis data dari aplikasi yang dibangun.



Gambar. 6. Basis Data

Adapun implementasi tiap tabel yang ada pada basis data dapat dilihat pada Gambar 7-10. Tabel hubungan sebagaimana yang ditunjukan pada Gambar 7 adalah tabel yang berisikan informasi terkait hubungan antar *node* pada aplikasi.

#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Komentar	Ekstra
1	id_hubungan	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	awal	int(255)			Tidak	*Tidak ada*		
3	tujuan	int(255)			Tidak	*Tidak ada*		
4	jarak	int(11)			Tidak	*Tidak ada*		

Gambar.7. Tabel Hubungan

Tabel *Node* yang ditunjukkan oleh Gambar 8 merupakan tabel yang berisi informasi *node* pada aplikasi.

#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Komentar	Ekstra
1	id_node	int(11)			Tidak	*Tidak ada*		AUTO_INCREMENT
2	lat	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		
3	lng	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		

Gambar. 8. Tabel *Node*

Gambar 9 menunjukkan Tabel Pengguna yang merupakan tabel yang berisi informasi pengguna pada aplikasi. Dan Tabel yang berisikan informasi tambal ban ditunjukkan oleh Gambar 10.

#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Komentar	Ekstra
1	id_pengguna	varchar(50)	utf8mb4_general_ci		Tidak	*Tidak ada*		
2	nama_lengkap	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		
3	nama_pengguna	varchar(150)	utf8mb4_general_ci		Tidak	*Tidak ada*		
4	kata_sandi	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		

Gambar. 9. Tabel Pengguna

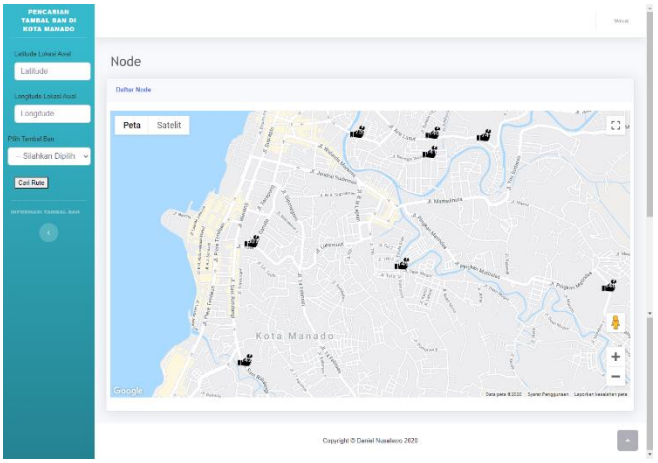
#	Nama	Jenis	Penyortiran	Atribut	Kosong	Bawaan	Komentar	Ekstra
1	id_tambal	varchar(150)	utf8mb4_general_ci		Tidak	*Tidak ada*		
2	nama_usaha	varchar(150)	utf8mb4_general_ci		Tidak	*Tidak ada*		
3	kategori	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		
4	alamat	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		
5	jam_operasional	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		
6	no_kontak	varchar(50)	utf8mb4_general_ci		Tidak	*Tidak ada*		
7	foto	varchar(255)	utf8mb4_general_ci		Tidak	*Tidak ada*		
8	id_node	int(11)			Tidak	*Tidak ada*		

Gambar. 10. Tabel Tambal Ban

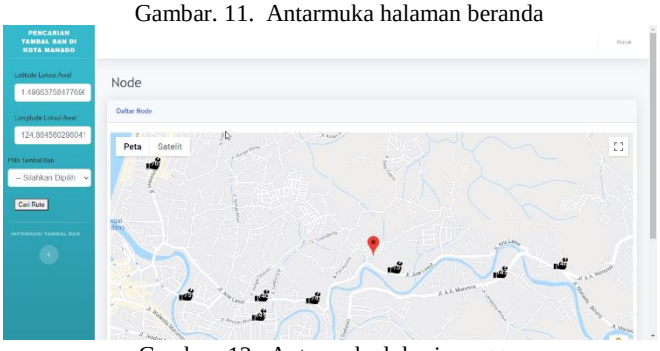
Pengodean Algoritma Dijkstra adalah sebagai berikut:

```
…
  {
    $index = 0;
    $dikunjungi = [];
    $sebelumnya = [];
    $jarak = [];
    $jarak_sementara = [];
    $query_node = $this->db->query("select lat , lng from
    node where id_node = '$awal'")->row();
    $dikunjungi[$awal] = 1;
    $jarak[$awal] = 0;
    $sementara = $awal;
    $sebelumnya[$awal] = $awal;

    $hasil = $this->db->query("SELECT hb.awal , hb.tujuan ,
    hb.jarak, tj.id_node , tj.lat , tj.lng, ROUND(111.111 *
    DEGREES(ACOS(LEAST(1.0, COS(RADIANS($query_node->lat))
         * COS(RADIANS(tj.Lat))
         * COS(RADIANS($query_node->lng - tj.Lng))
         + SIN(RADIANS($query_node->lat))
          * SIN(RADIANS(tj.Lat))))) * 1000 , 2)  AS
    jarak_dari FROM hubungan as hb left join node as tj on
    hb.tujuan = tj.id_node order by jarak_dari")->result();

    $hubungan = $this->buat_hubungan($hasil, $awal,
    $tujuan);

    $perbandingan = [];

    foreach ($hubungan as $node_asal => $tetangga) {
      foreach ($tetangga as $id_tetangga => $cost) {
      if (!isset($dikunjungi[$node_asal]) ||
      !isset($dikunjungi[$id_tetangga])) {
           if (!isset($jarak[$id_tetangga]) && $sementara
           == $awal) {
              $jarak[$id_tetangga] = $cost;
              $sebelumnya[$id_tetangga] = $sementara;
           }
           if ($sementara !== $awal) {
              if (!isset($jarak[$id_tetangga])) {
                 $sebelumnya[$id_tetangga] = $sementara;
                 $jarak[$id_tetangga] =
                 $jarak[$sebelumnya[$id_tetangga]] + $cost;
               } else {
               if ($jarak[$id_tetangga] + $cost <
               $jarak[$id_tetangga]) {
                    $sebelumnya[$id_tetangga] = $sementara;
                    $jarak[$id_tetangga] =
                    $jarak[$id_tetangga] + $cost;
                }
              }
           }
      }

      $dipilih = array_search(min($perbandingan),
      $perbandingan);
      $dikunjungi[$dipilih] = 1;

      $sementara = $dipilih;
      unset($perbandingan[$dipilih]);
      if ($dipilih == $tujuan) {
        break;
      }
    }

    $hasil = new stdClass();
    $hasil->rute = $sebelumnya;
    $hasil->jarak = $jarak;
    return $hasil;
  }
…
```

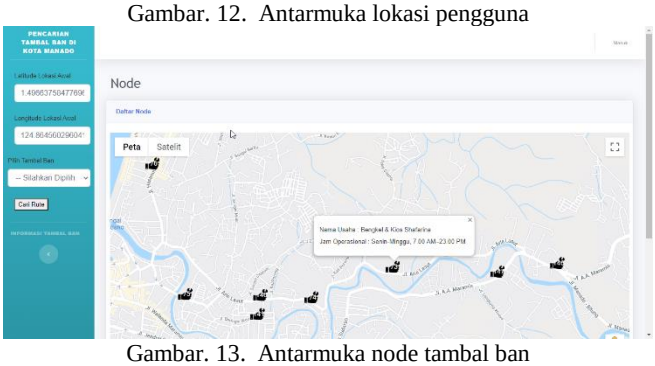
Implementasi Antarmuka ditunjukkan pada Gambar 11 – Gambar 22.



Gambar. 11. Antarmuka halaman beranda



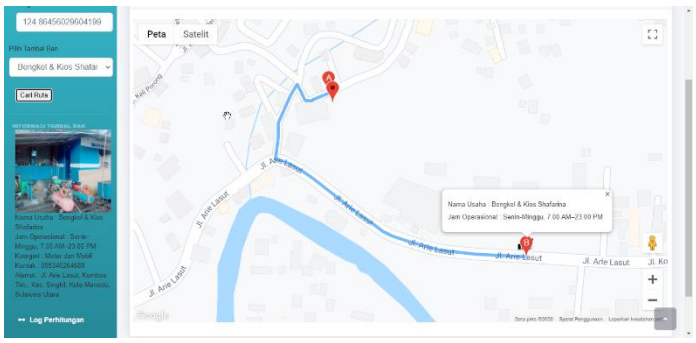
Gambar. 12. Antarmuka lokasi pengguna



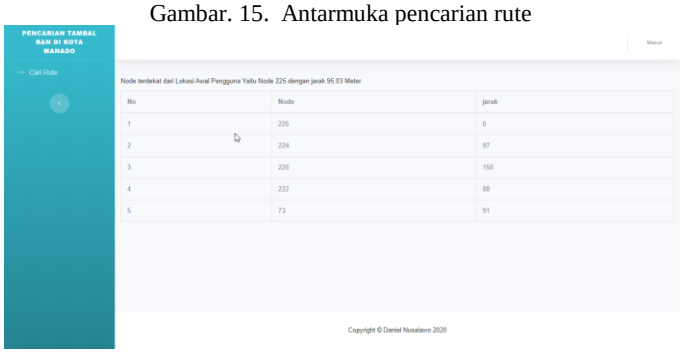
Gambar. 13. Antarmuka node tambal ban

Gambar. 14. Antarmuka pemilihan tambal ban

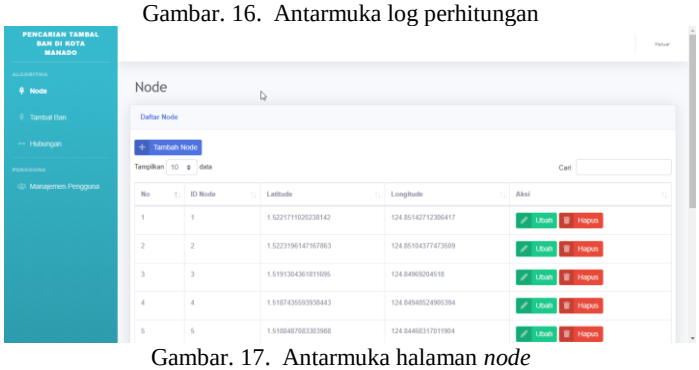
Tampilan untuk memilih tambal ban yang dituju yang ditunjukkan pada Gambar 14 merupakan tampilan saat pengguna melakukan pilih tambal ban pada form yang ada pada pojok kiri aplikasi, sehingga di bawahnya muncul informasi detail tambal ban seperti gambar, nama usaha, jam operasional, kategori tambal ban, kontak dan alamat.



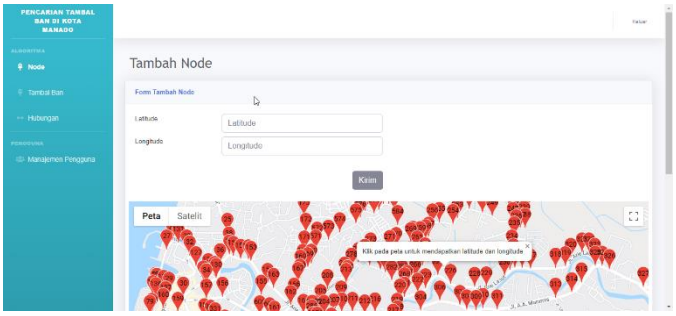
Gambar. 15. Antarmuka pencarian rute



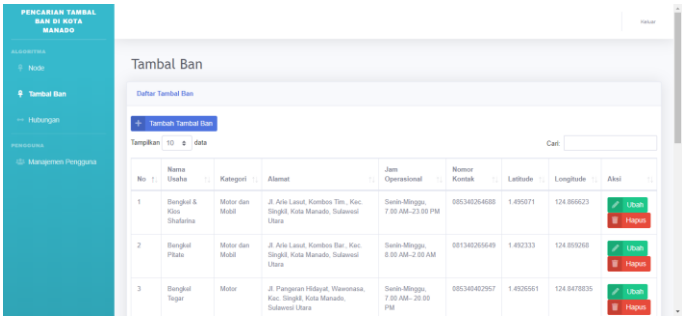
Gambar. 16. Antarmuka log perhitungan



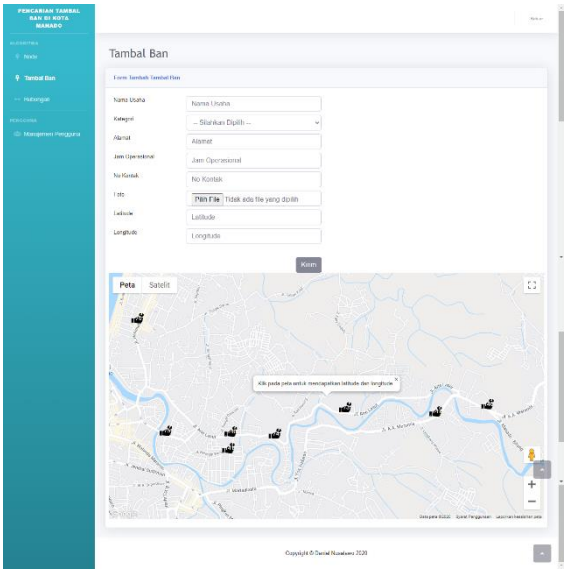
Gambar. 17. Antarmuka halaman *node*



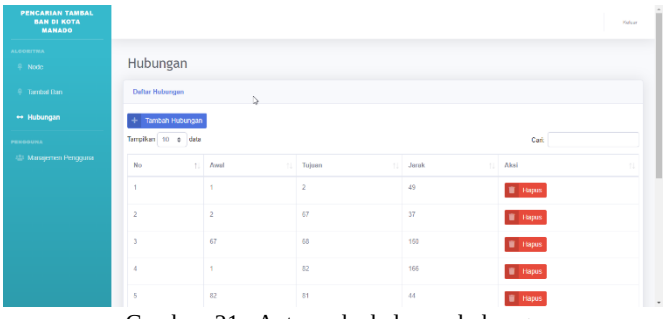
Gambar. 18. Antarmuka halaman tambah *node*



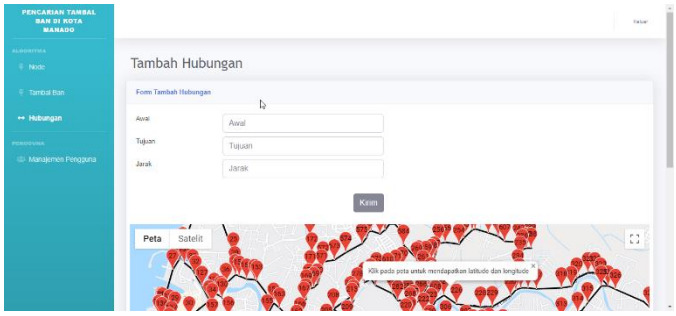
Gambar. 19. Antarmuka halaman tambal ban



Gambar. 20. Antarmuka halaman tambah tambal ban



Gambar. 21. Antarmuka halaman hubungan

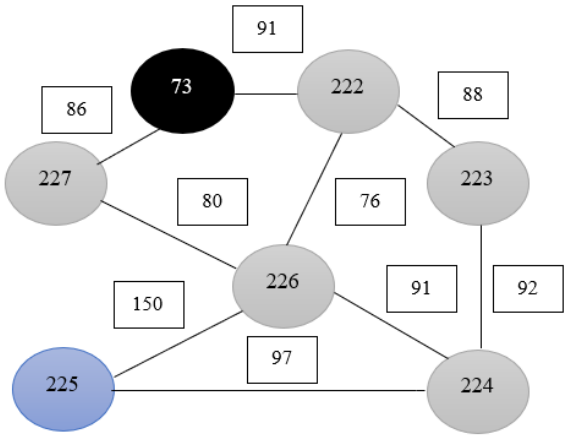


Gambar. 22. Antarmuka halaman tambah hubungan

Pengujian

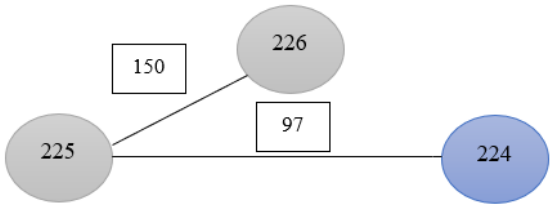
Pengujian dilakukan untuk mengetahui apakah aplikasi yang dibangun sudah sesuai dengan rancangan yang dibuat sebelumnya khususnya pada perhitungan Algoritma Dijkstra. Berikut ini penjabaran langkah demi langkah yang terperinci untuk menemukan rute terdekat dari titik awal ke titik target dengan bobot jarak terkecil.

1. Titik awal 225, titik target 73. Setiap sisi yang terhubung antar titik memiliki nilai.



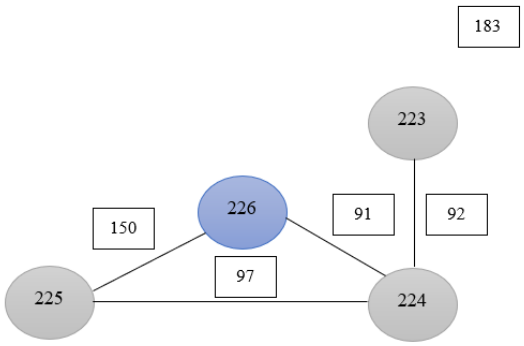
Gambar. 23. Langkah 1

2. Dijkstra menghitung titik tetangga (titik 225) yang terhubung langsung ke titik keberangkatan, dan hasil yang diperoleh yaitu titik 224 karena dibandingkan dengan nilai titik lain, titik 224 memiliki bobot terkecil, nilai = (0+97) = 97.



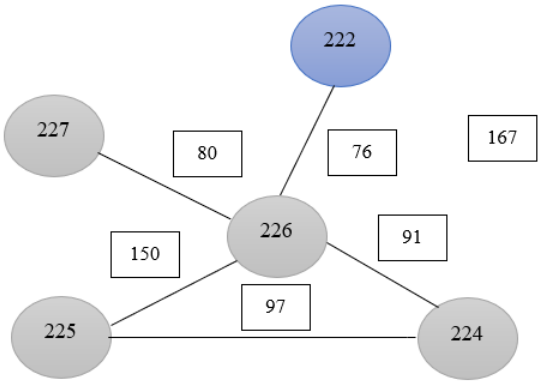
Gambar. 24. Langkah 2

3. Titik 224 di atur sebagai titik awal lalu ditandai menjadi titik yang dapat disentuh. Dijkstra menghitung ulang titik tetangga yang terhubung langsung ke titik yang disentuh. Perhitungan Dijkstra menunjukkan bahwa titik 226 menjadi titik keberangkatan berikutnya sebab bobotnya kurang dari nilai perhitungan terakhir = (0 + 91) = 91.



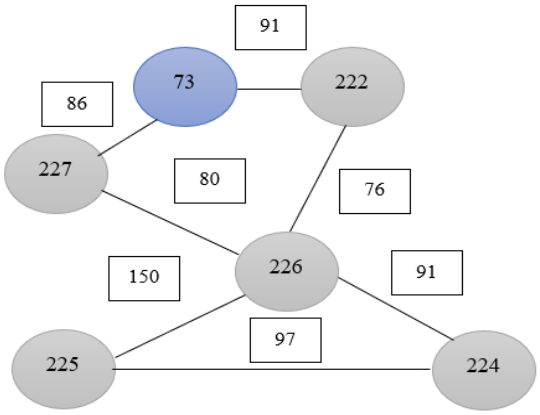
Gambar. 25. Langkah 3

4. Titik 226 ditandai sebagai titik yang sudah disentuh melanjutkan perhitungan. Dari seluruh titik yang berdekatan yang belum tersentuh yang terhubung langsung ke titik yang disentuh, titik berikutnya yang ditandai sebagai titik tersentuh yaitu titik 222, sebab nilai bobotnya adalah yang paling kecil = (91+76) = 167.



Gambar. 26. Langkah 4

5. Titik 222 menjadi titik yang dapat disentuh, Dijkstra menghitung ulang dan menemukan bahwa ia telah mencapai titik 73 (titik target) melalui titik 222. Jalur terpendek adalah 225-226-222-73, nilai bobot yang diperoleh adalah (167 + 91) = 258 Setelah mencapai titik target, perhitungan Dijkstranya dinyatakan selesai.



Gambar. 27. Langkah 5

V.KESIMPULAN DAN SARAN

Adapun Kesimpulan yang diperoleh, yaitu:

1. Aplikasi pencarian rute terpendek tambal ban di Kota Manado dengan menggunakan Algoritma Dijkstra telah berhasil dibangun.
2. Algoritma Dijkstra yang diterapkan sudah dapat memberikan rute terpendek pada pencarian lokasi tambal ban di Kota Manado.
3. Dengan adanya aplikasi ini, pengguna kendaraan bermotor dapat dengan mudah mendapatkan informasi lokasi tambal ban di Kota Manado.

Adapun Saran bagi pengembangan selanjutnya, yaitu :

1. Aplikasi dapat dikembangkan pada *platform* yang lain
2. Dalam pengembangan selanjutnya, aplikasi dapat membaca lokasi awal pengguna secara *realtime* dan terdapat fitur navigasi untuk mempermudah menuju lokasi tambal ban yang dipilih.

DAFTAR PUSTAKA

[1] Ni. Retnowati and R. S. Lutfiyani, "PERBANDINGAN ALGORITMA DJIKSTRA DAN WARSHALL DALAM PENENTUAN LINTASAN TERPENDEK KE KOTA KLATEN," *Unisda J. Math. Comput. Sci.*, vol. 4, no. 2, 2018.

[2] Sutriono, "IMPLEMENTASI ALGORITMA DIJKSTRA PADA APLIKASI PENCARIAN BENGKEL TAMBAL BAN TERDEKAT DI KOTA MEDAN BERBASIS ANDROID," *J. Maj. Ilm. Inf. dan Teknol. Ilm.*, vol. 7, no. 1, pp. 41–45, 2019.

[3] Fitria and A. Triansyah, "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] W. O. A. P. Wulandari, B. Pramono, and L. Tajidun, "APLIKASI PENCARIAN RUTE TERPENDEK APOTEK DI KOTA KENDARI MENGGUNAKAN ALGORITMA FLOYD-WARSHALL," *semanTIK*, vol. 3, no. 1, 2017.

[5] A. S. Rossa and M. Shalahuddin, *Rekayasa Perangkat Lunak Terstruktur dan Berorientasi Objek*. Bandung: Informatika, 2016.