BAB IV
HASIL DAN PEMBAHASAN

Pada bab ini akan membahas mengenai implementasi sistem dan hasil akhir dari penelitian yang sudah dilakukan, mulai dari implementasi desain antarmuka sistem dilanjutkan dengan implementasi perhitungan algoritma dijkstra dengan menggunakan bahasa pemrograman PHP dan Javascript sebagai bahasa utama dan MySQL sebagai sumber pengelolaan data. Terakhir yaitu pembahasan terhadap pengujian sistem yang telah dibuat.

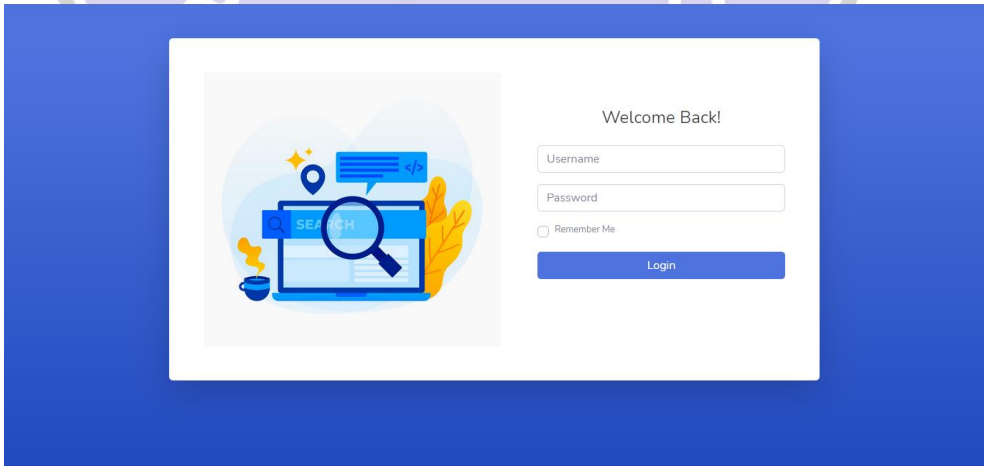
4.1 Hasil Implementasi Sistem

Implementasi yang dihasilkan dari desain sistem yang sudah dibahas pada bab 3 adalah sebagai berikut :

Interface Sistem :

Untuk halaman interface sistem yang telah dibuat antara Admin dan User kurang lebih hampir sama, perbedaan utama hanya terletak pada hak akses di setiap menu sistem yang ada. Admin mempunyai seluruh hak akses, sedangkan untuk user ada beberapa menu yang tidak ditampilkan sebagai pembatasan hak akses.

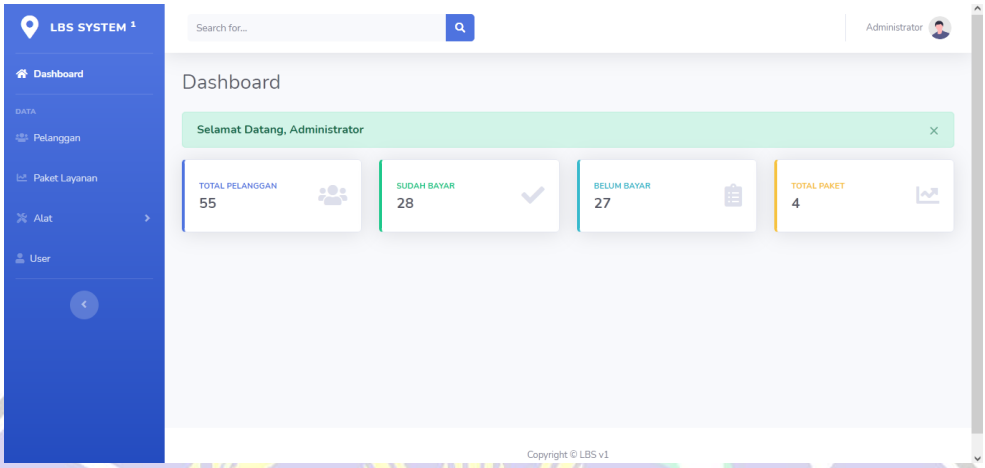
1. Interface Login



Gambar 4. 1 Interface Login

Halaman yang akan diakses pertama kali, untuk dapat login pengguna harus memasukkan username dan password yang teIah ditentukan, jika login berhasil akan masuk ke menu selanjutnya yaitu dashboard.

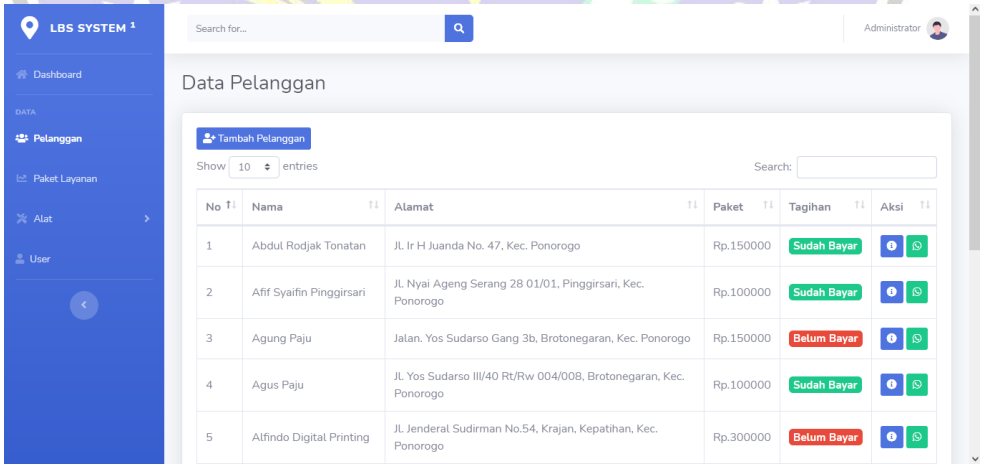
2. Interface Dashboard



Gambar 4. 2 Interface Dashboard

Pada Tampilan Dashboard pengguna dapat melihat informasi umum yang ditampilkan seperti jumlah total pelanggan, jumlah pelanggan yang sudah bayar tagihan, jumlah pelanggan belum bayar tagihan dan jumlah paket.

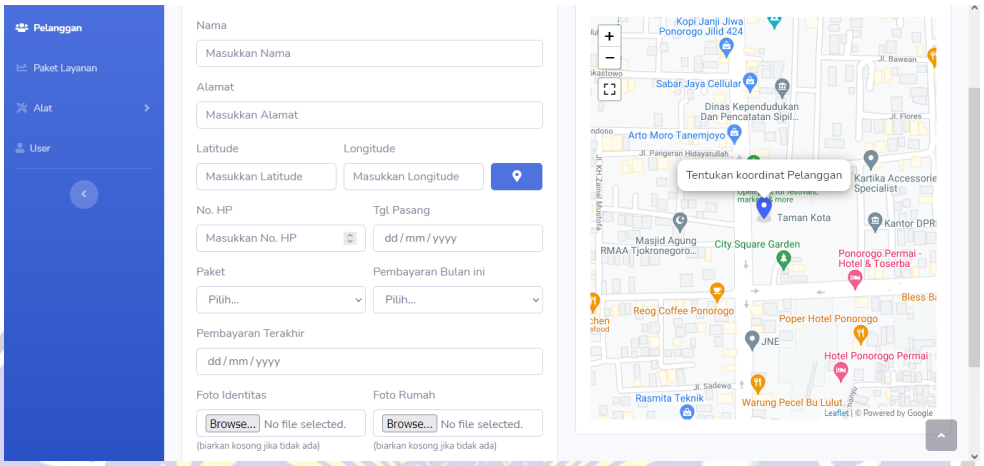
3. Interface Menu Pelanggan



Gambar 4. 3 Interface Menu Pelanggan

Didalam menu ini terdapat beberapa informasi dari pelanggan diantaranya yaitu nama, alamat, paket layanan, informasi tagihan, dan terdapat 2 tombol yaitu tombol detail dan tombol whatsapp untuk memberikan pemberitahuan terkait tagihan layanan.

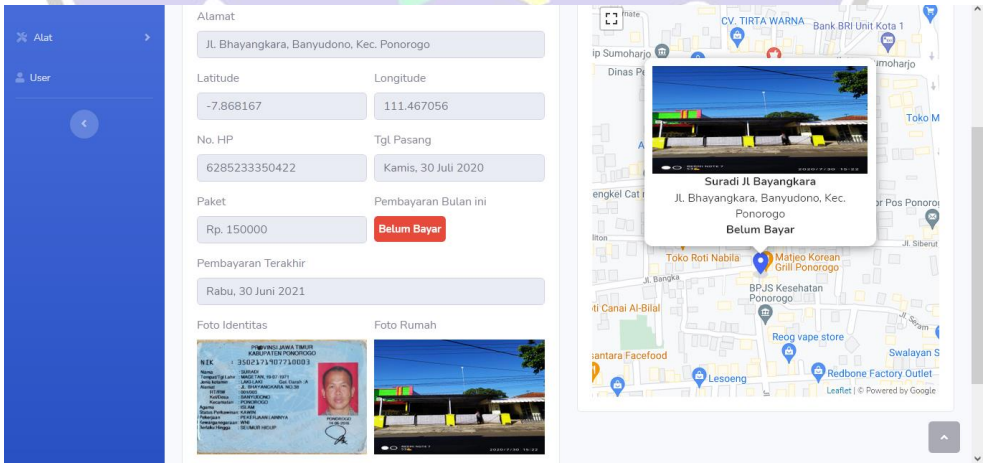
4. Interface Tambah Pelanggan



Gambar 4. 4 Interface Tambah Pelanggan

Pada menu ini admin atau petugas dapat menambahkan data pelanggan baru. Data yang harus di inputkan meliputi nama, alamat, latitude, longitude, no.hp, tanggal pemasangan, paket layanan, informasi pembayaran, foto identitas dan foto rumah.

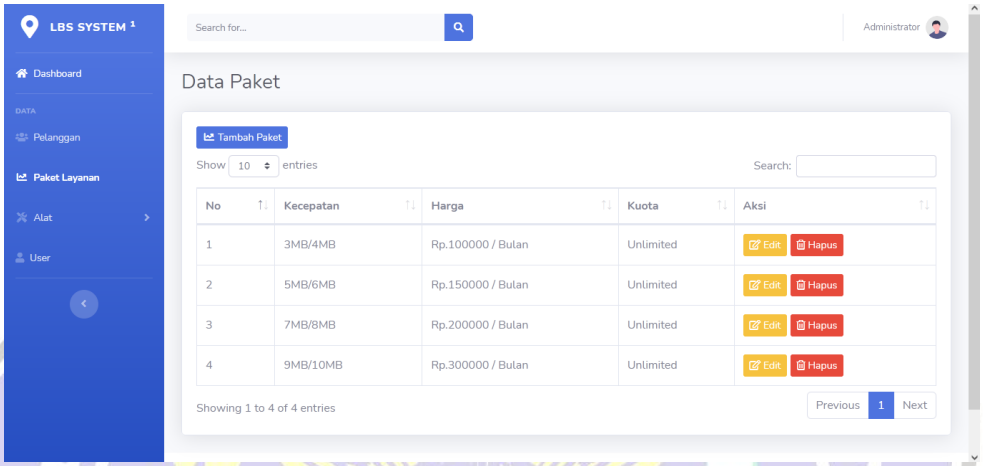
5. Interface Detail Pelanggan



Gambar 4. 5 Interface Detail Pelanggan

Interface detail pelanggan akan menampilkan seluruh informasi dari pelanggan tersebut, data yang ditampilkan adalah nama, alamat, latitude, longitude, no.hp, tanggal pemasangan, paket layanan, informasi pembayaran, foto identitas dan foto rumah.

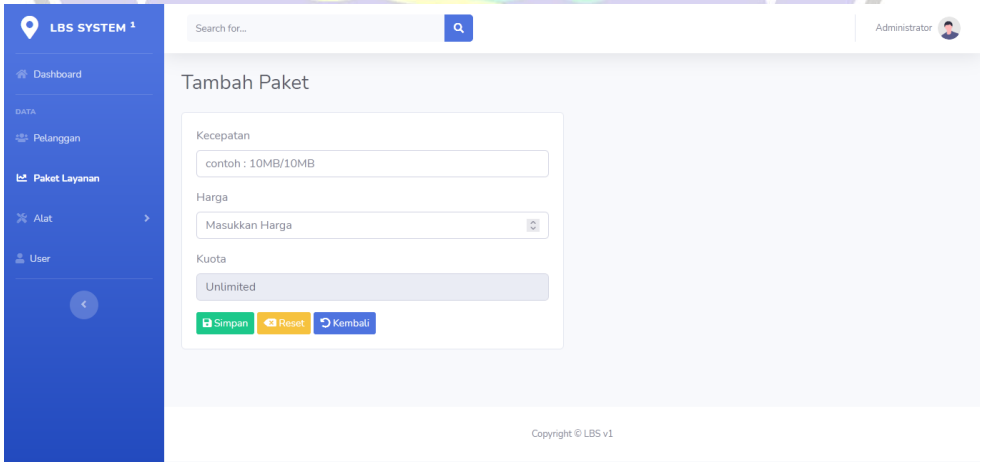
6. Interface Menu Peket



Gambar 4. 6 Interface Menu Paket

Didalam menu ini terdapat beberapa informasi dari paket layanan yang ditawarkan diantaranya yaitu kecepatan internet, harga, dan kuota yang didapat.

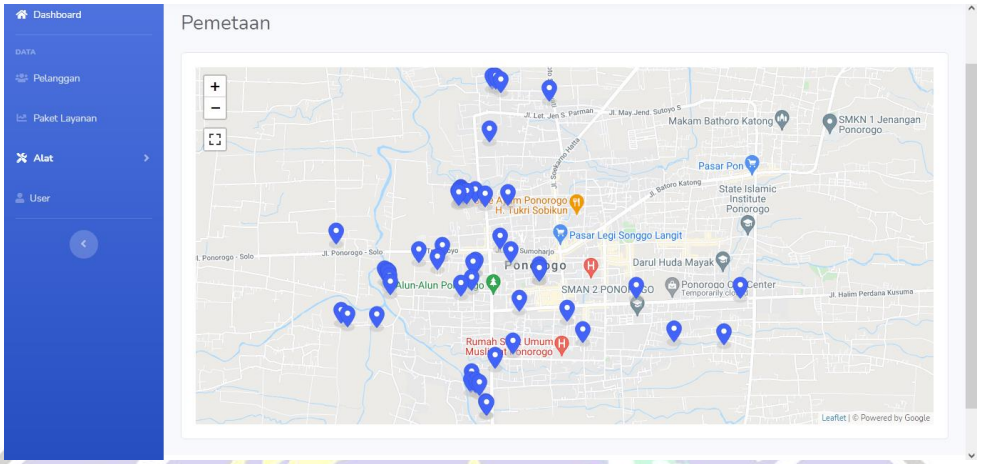
7. Interface Tambah Paket



Gambar 4. 7 Interface Tambah Paket

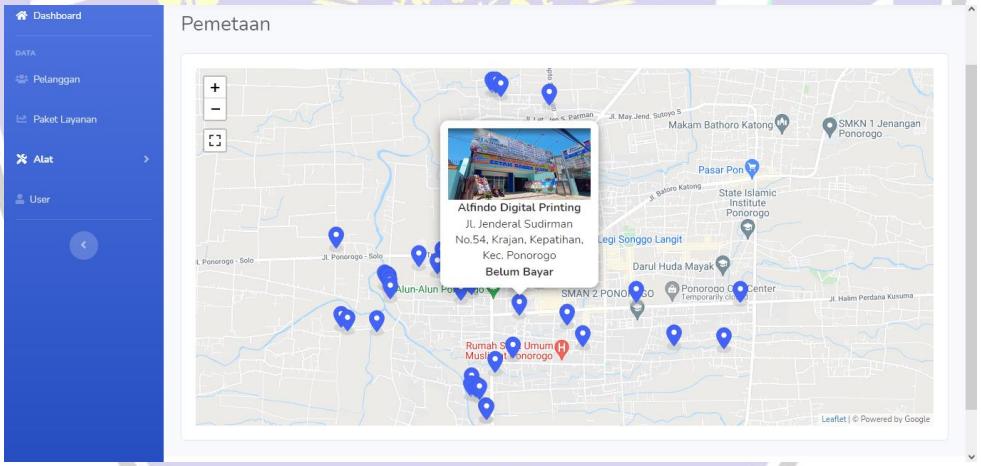
Didalam menu ini admin atau petugas dapat menambahkan data paket baru. Data yang harus di inputkan meliputi kecepatan layanan dan harga paket.

8. Interface Menu Pemetaan



Gambar 4. 8 Interface Menu Pemetaan

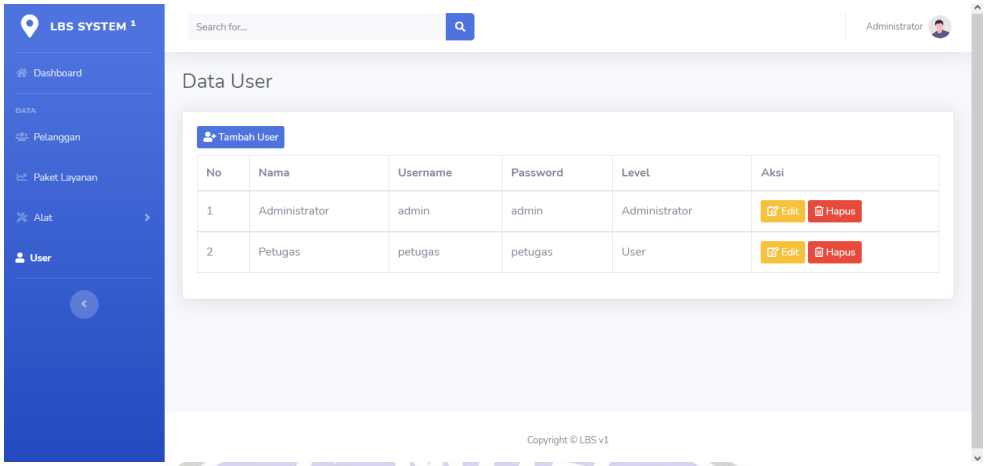
Menu ini akan menampilkan sebaran lokasi pelanggan dalam peta.



Gambar 4. 9 Interface Popup Menu Pemetaan

Saat salah satu node di klik, maka akan memunculkan popup yang memberikan informasi pelanggan dari lokasi tersebut. Data yang di tampilkan antara lain nama, alamat, dan status pembayaran.

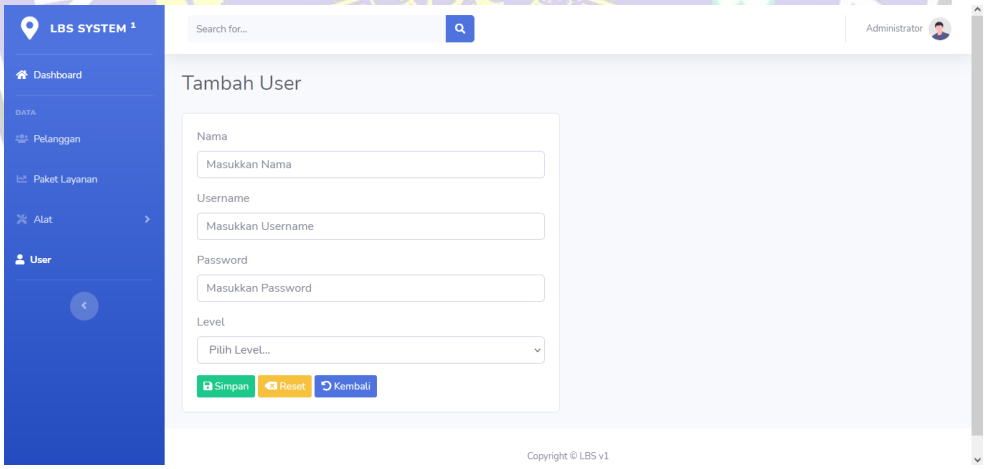
9. Interface Menu User



Gambar 4. 10 Interface Menu User

Menu ini akan menampilkan pengguna yang dapat mengakses sistem, berisi nama, username, password, dan level. Menu ini hanya dapat diakses oleh admin, sedangkan untuk user menu di akan di sembunyikan untuk membatasi hak akses.

10. Interface Tambah User



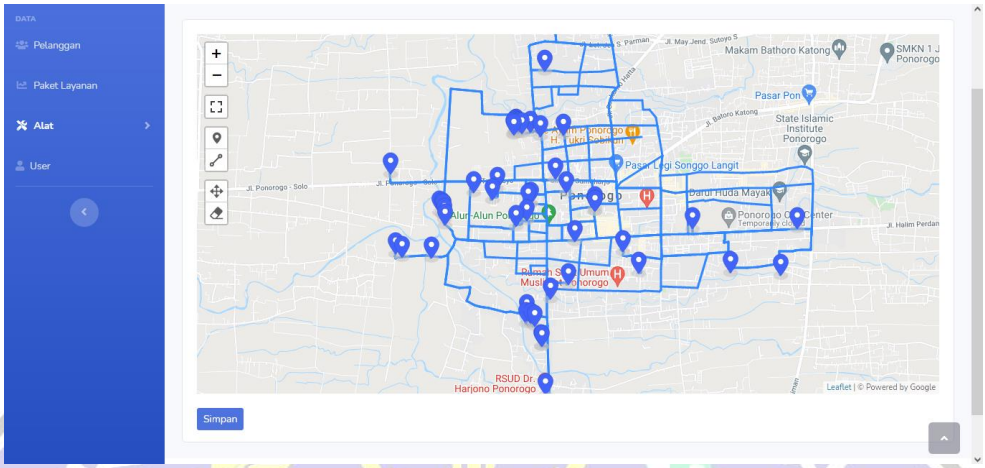
Gambar 4. 11 Interface Tambah User

Pada menu ini admin dapat menambahkan pengguna baru. Data yang harus di inputkan meliputi nama, username, password, dan level.

4.2 Implementasi Algoritma Dijkstra Pada Sistem

Implementasi yang dihasilkan dari penerapan algoritma kedalam sistem yang sudah dibahas pada bab 3 adalah sebagai berikut :

1. Interface Pengaturan Node dan Graf



Gambar 4. 12 Interface Pengaturan Node dan Graf

Proses pembuatan node dengan menentukan titik koordinat untuk rute jalan dan titik-titik lokasi pelanggan. Setelah penentuan node selesai kemudian membuat graf, untuk menghubungkan simpul-simpul yang telah dibuat. Simpul digunakan sebagai tanda persimpangan jalan, lokasi pelanggan dan graf merupakan panjang jalan antara simpul. Bobot setiap graf akan dihitung oleh sistem. Bobot yang akan ditambahkan dihitung dari jarak antar simpul.. Dibawah ini adalah source code dalam pembuatan graf pada sistem.

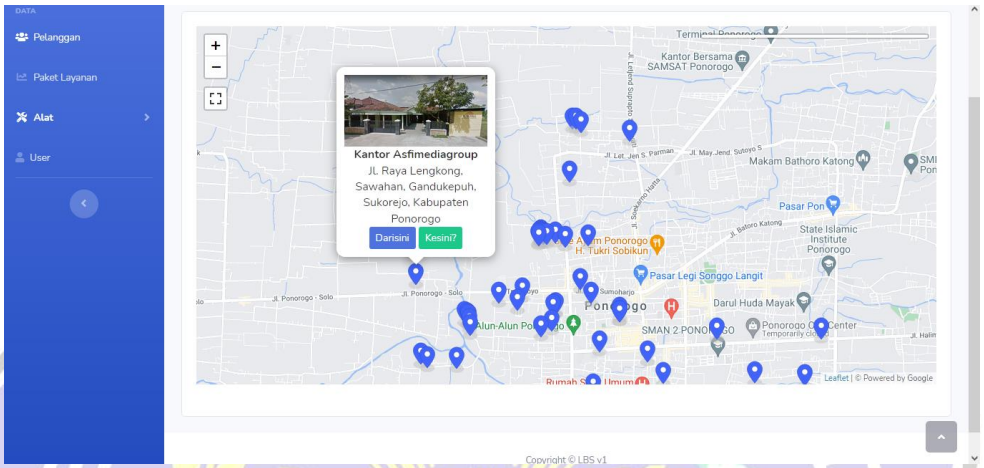
```
var graph_data = [];
    function createGraphData(data) {
        if (data.length > 0) {
            for (i = 0; i < (data.length - 1); i++) {
                graph_data.push({
                    start_lat: data[i].lat,
                    start_long: data[i].long,
                    end_lat: data[(i + 1)].lat,
                    end_long: data[(i + 1)].long,
                    distance: (map.distance([data[i].lat, data[i].long],
                    [data[(i + 1)].lat, data[(i + 1)].long]) / 1000),
                });
            }
```

```
            $.ajax({
                url: '<?= site_url('node/saveGraphAjax') ?>',
                data: {
                    'data': graph_data
                },
                type: 'POST',
                success: function(e) {
                    window.location.reload();
                }
            })
        }
    }
    function generateGeoJson() {
        var vertex_data = [];
        var fg = L.featureGroup();
        var layers = findLayers(map);
        layers.forEach(function(layer) {
            if (layer._drawnByGeoman == true) {
                fg.addLayer(layer);
            }
        });
        fg.toGeoJSON().features.forEach(function(e) {
            if (e.geometry.type == 'LineString') {
                e.geometry.coordinates.forEach(function(coor) {
                    data = {
                        'long': coor[0],
                        'lat': coor[1],
                    };
                    vertex_data.push(data);
                })
            }
        })
        createGraphData(vertex_data);
        if (remove_line.length > 0) {
            removeLine();
        }
    }
    function findLayers(map) {
        var layers = [];
        map.eachLayer(layer => {
            if (
                layer instanceof L.Polyline || layer instanceof L.Marker
                ) {layers.push(layer);}
        });
```

```
        layers = layers.filter(layer => !!layer.pm);
        layers = layers.filter(layer => !layer._pmTempLayer);
        return layers;
    }
```

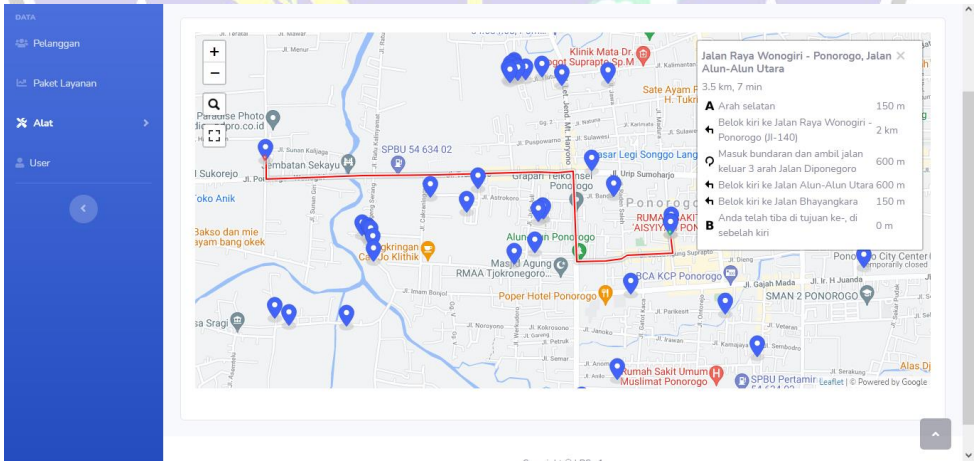
Gambar 4. 13 Source Code Pembuatan Graf

2. Interface Pencarian Rute



Gambar 4. 14 Interface Pencarian Rute

Untuk mencari rute terdekat maka pengguna harus menentukan posisi awal dan lokasi tujuan. Setelah posisi awal dan posisi tujuan telah ditentukan maka, Tahap selanjutnya algoritma dijkstra akan menghitung rute rute yang akan dilewati dengan nilai jarak terkecil.



Gambar 4. 15 Interface Hasil Pencarian Rute

Dibawah ini adalah source code perhitungan algoritma dijkstra pada sistem.

```php
<?php
class Djikstra
{
    public $graf;
    public $grafName;
    public $lok_asal;
    public $lok_akhir;
    public $visited_node = array();
    public $current_node = '';
    public $shortest_distance = array();
    public $previous_node = array();

    /**
     * __construct
     *
     * @param  mixed $graf
     * @param  mixed $grafName
     * @param  mixed $lok_asal
     * @param  mixed $lok_akhir
     * @return void
     * pada fungsi ini, kita akan melewatkan $graf,$graname,$lok_asal,
       dan $lok_akhir
     * selanjutnya, kita perlu untuk membuat matrix dimana
       indexnya merupkan node awal ke node akhir berasal dari grafname dengan
     * nilai adalah jarak antar node
     */
    public function __construct($graf = '', $graName = '', $lok_asal = '',
                                $lok_akhir = '')
    {
        $this->lok_asal = $lok_asal;
        $this->lok_akhir = $lok_akhir;
        $this->graf = $graf;
        $this->grafName = $grafName;

        foreach ($this->grafName as $ki => $vi) {
            foreach ($this->grafName as $kj => $vj) {
                if (isset($this->graf["$ki"]["$kj"])) {
                    $this->graf["$ki"]["$kj"] = $this->graf["$ki"]["$kj"];
                } else {
                    $this->graf["$ki"]["$kj"] = INF;
                }
            }
        }
        /**
         * kita menjalankan algoritma
         */
        $this->run();
```

```
}
/**
 * run
 *
 * @return void
 * ini adalah fungsi utama dimana algoritma akan dijalankan
 */
public function run()
{
    $no = 0;
    while (true) {

        if ($this->current_node == '') {

            // Start dengan set node awal (A) sebagai current node.
            $this->current_node = $this->lok_asal;
            $this->shortest_distance[$this->current_node] = 0;
        }
        // Periksa semua node yang terhubung ke A dan perbarui "Jarak dari
           A" dan atur "node sebelumnya" ke "A".
        $checkAllNodes = $this->checkAllNodes($this->current_node);
        /**
         * jika node tersedia
         */
        if ($checkAllNodes) {
            /**
             * lakukan perulangan dimana jarak/distance tidak sama dengan INF
             */
            foreach ($checkAllNodes as $key => $val) {

                if ($val != INF) {
                    if (!in_array($key, $this->visited_node)) {
                        if (isset($this->shortest_distance[$key])) {
                            /**
                             * JIka jarak current node ditambah dengan jarak
                               yang baru lebih kecil dari jarak yang lama
                             * maka jarak saat ini diperbarui
                             */
                            if (($this->shortest_distance[$this->
                               current_node] + $val) < $this->
                                shortest_distance[$key]) {
                                $this->shortest_distance[$key] = $this->
                                 shortest_distance[$this->
                                 current_node] + $val;
                                $this->previous_node[$key] = $this->
                                 current_node;
                            }
```

```
                } else {
                    $this->shortest_distance[$key] = $this->
                            shortest_distance[$this->
                            current_node] + $val;
                    $this->previous_node[$key] = $this->current_node;
                }
            }
        }
    }

    /**
     * masukan current node ke dalam daftar dikunjungi
     */
    array_push($this->visited_node, $this->current_node);


    $prev = array_filter($this->previous_node, function ($e) {
        return $e == $this->current_node;
    });

    $prev = array_keys($prev);
    $temp = array();
    for ($i  = 0; $i < count($prev); $i++) {

        $temp[$prev[$i]] = $this->shortest_distance[$prev[$i]];
    }
    if (count($temp) > 0) {
        $min = array_keys($temp, min($temp));
        if (count($min) > 0) {
            $this->current_node = $min[0];
        } else {
            // ----------------------
            break;
        }
    } else {

        $temp = array();
        foreach ($this->shortest_distance as $k => $v) {
            if (!in_array($k, $this->visited_node)) {
                $temp[] = $k;
            }
        }
        $sisa_node_yang_blm_dikunjungi = array();

        for ($i  = 0; $i < count($temp); $i++) {

            $sisa_node_yang_blm_dikunjungi[$temp[$i]] = $this->
```

```
                    shortest_distance[$temp[$i]];
                }
                if (count($sisa_node_yang_blm_dikunjungi) > 0) {
                    $min = array_keys($sisa_node_yang_blm_dikunjungi,
                           min($sisa_node_yang_blm_dikunjungi));
                    if (count($min) > 0) {
                        $this->current_node = $min[0];
                    }
                }
            }
        } else {
            // break;
        }

        /**
         * jika current node/atau node saat ini sama dengan lok akhir,
           maka proses pencarian dihentikan
         */

        if ($this->current_node == $this->lok_akhir) {
            break;
        }

        $no++;
    }
}

/**
 * checkAllNodes
 *
 * @param  mixed $node
 * @return void
 * fungsi ini digunakan untuk memeriksa suksesor, atau membuka semua node
   yang terkait dengan best node
 */
public function checkAllNodes($node)
{
    if (array_key_exists($node, $this->graf)) {
        return $this->graf[$node];
    }
    return false;
}

/**
 * getPath
 *
 * @return void
```

```
     * fungsi ini digunakan untuk mendapatkan path atau jalur yang tersedia
       untuk dilewati dari lokasi awal dan lokasi akhir
     */
    public function getPath()
    {
        $node = $this->previous_node;
        $temp = '';
        $path = array();
        while (true) {
            if ($temp == '') $temp = $this->lok_akhir;
            array_push($path, $temp);
            if (!isset($node[$temp])) break;
            $temp = $node[$temp];
        }
        return $path;
    }
    /**
     * getDistance
     *
     * @return void
     * fungsi ini digunakan untuk mendapatkan jarak dari lokasi awal ke lokasi
       akhir
     */
    public function getDistance()
    {
        $path = $this->getPath();
        $path = array_reverse($path);
        $total = 0;
        for ($i = 0; $i < (count($path) - 1); $i++) {
            $total += $this->graf[$path[$i]][$path[($i + 1)]];
        }
        return $total;
    }
}
```

Gambar 4. 16 Source Code Penerapan Algoritma Dijkstra

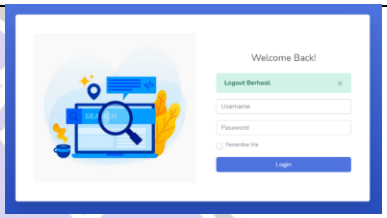
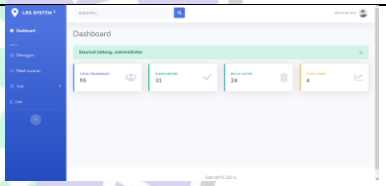
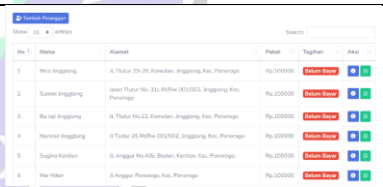
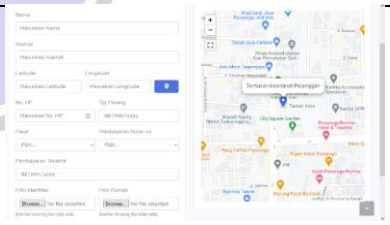
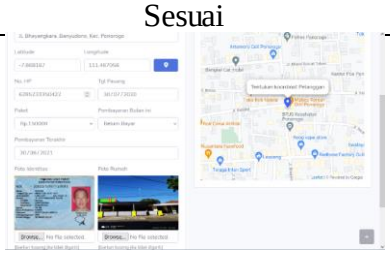
4.3 Pengujian Sistem

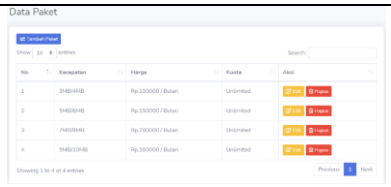
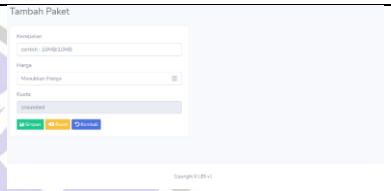
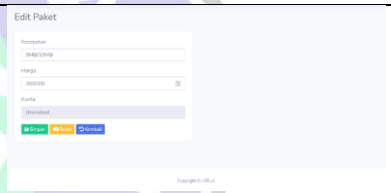
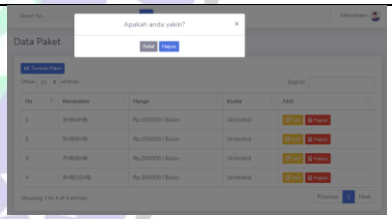
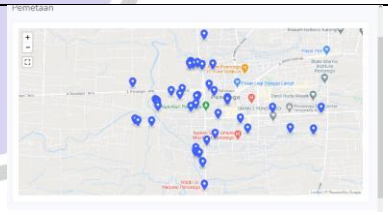
Pengujian sistem merupakan tahapan akhir dimana sistem yang baru diuji sehingga dapat diketahui kekurangan atau kelemahan sistem yang nantinya dapat dilakukan evaluasi dan perbaikan terhadap sistem. Pengujian dilakukan oleh pengguna aplikasi, selain untuk mengetahui kelemahan sistem pengujian ini juga digunakan untuk mengetahui tingkat keakurasian dan

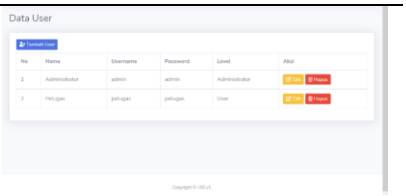
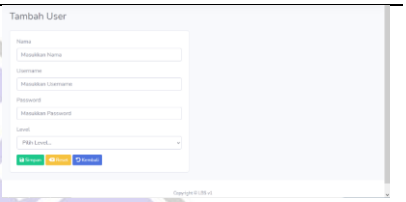
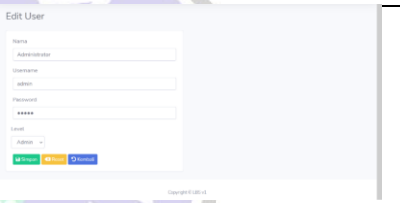
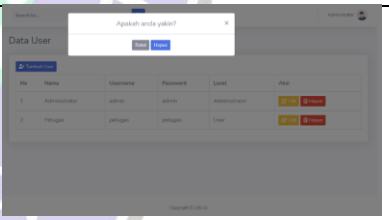
ketepatan informasi yang dihasilkan, sehingga sesuai dengan kebutuhan pengguna yang diharapkan.

4.3.1 Pengujian Blackbox

Pengujian ini dilakukan untuk memastikan setiap fitur pada sistem bekerja dengan sesuai tanpa kesalahan, pengujian dilakukan dengan metode blackbox testing. Hasil dari pengujian blackbox ini dapat dilihat pada tabel dibawah ini :

No	Test Case	Input	Expected Result	Actual Result
1.	Masuk ke browser, alamat web : http://localhost/lbs	Memasukkan alamat ke url pencarian	Menampilkan Halaman login	 Sesuai
2.	Login	Mengisi email dan password	Menampilkan Halaman Dashboard	 Sesuai
3.	Melihat daftar pelanggan	Klik menu, pilih pelanggan	Menampilkan data pelanggan dan aksi detail dan kirim notifikasi	 Sesuai
4.	Menambahkan data pelanggan baru	Klik menu pelanggan, pilih tambah pelanggan	Menampilkan formulir tambah pelanggan	 Sesuai
5.	Mengedit data pelanggan	Klik menu pelanggan, pilih detail, pilih edit	Menampilkan formulir edit pelanggan	 Sesuai

6.	Menghapus data pelanggan	Klik menu pelanggan, pilih detail, pilih hapus	Menampilkan popup konfirmasi penghapusan data pelanggan		Sesuai
7.	Melihat daftar paket	Klik menu, pilih paket	Menampilkan data paket		Sesuai
8.	Menambahkan data paket baru	Klik menu paket, pilih tambah paket	Menampilkan formulir tambah paket		Sesuai
9.	Mengedit data paket	Klik menu paket, pilih edit	Menampilkan formulir edit paket		Sesuai
10.	Menghapus data paket	Klik menu paket, pilih hapus	Menampilkan popup konfirmasi penghapusan data paket		Sesuai
11.	Melihat pemetaan pelanggan	Klik menu alat, pilih pemetaan	Menampilkan sebaran data pelanggan kedalam peta		Sesuai
12.	Mengatur node	Klik menu alat, pilih setting node	Menampilkan node dan graf kedalam peta		Sesuai

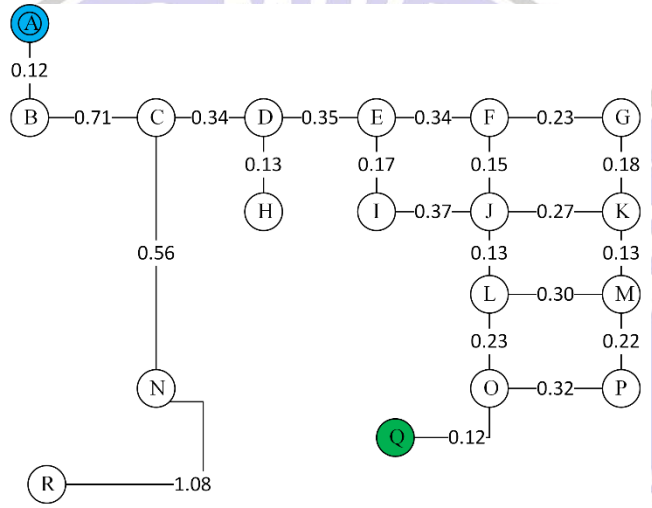
13.	Pencarian rute	Klik menu alat, pilih pencarian rute	Menampilkan graf dan rute yang dilalui kedalam peta		Sesuai
14.	Melihat daftar user	Klik menu, pilih user	Menampilkan data pengguna sistem		Sesuai
15.	Menambahkan data user baru	Klik menu user, pilih tambah user	Menampilkan formulir tambah user		Sesuai
16.	Mengedit data user	Klik menu user, pilih edit user	Menampilkan formulir edit user		Sesuai
17.	Menghapus data user	Klik menu user, pilih hapus	Menampilkan popup konfirmasi penghapusan data user		Sesuai

Tabel 4. 1 Hasil Pengujian Sistem

4.3.2 Pengujian Algoritma

Pengujian ini dilakukan untuk untuk mengetahui kinerja dari perhitungan algoritma dijkstra apakah dapat menentukan rute terpendek dalam mencari lokasi pelanggan dan menampilkannya kedalam peta dan sesuai dengan perhitungan yang dilakukan secara manual.

Pengujian pertama dilakukan dengan mencari rute dari kantor asfimediagroup yang beralamatkan Jl. Raya Lengkong, Gandukepuh, Sukorejo menuju ke lokasi pelanggan yang bernama Bapak Sudarsono dengan alamat Jl. Julajuli Menggunakan perhitungan algoritma dijkstra secara manual.



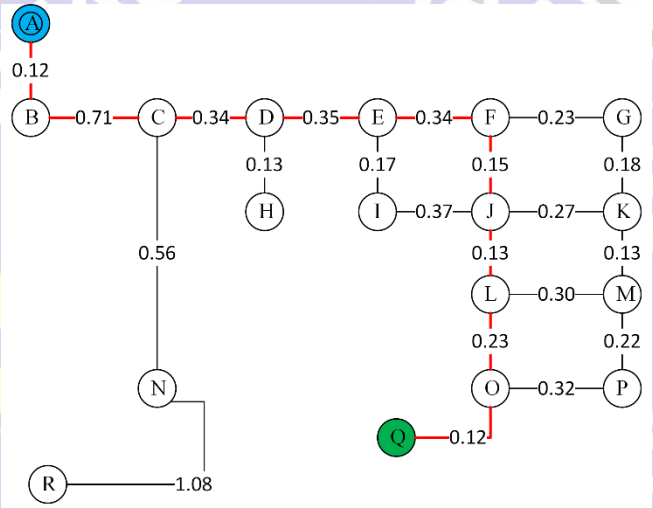
Gambar 4. 17 Permodelan Graf Pengujian Pertama

V	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
A	0A	0.12A	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
B		0.12A	0.83B	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
C			0.83B	1.17C	∞	∞	∞	∞	∞	∞	∞	∞	∞	1.39C	∞	∞	∞	∞
D				1.17C	1.52D	∞	∞	1.30D	∞	∞	∞	∞	∞	1.39C	∞	∞	∞	∞
H					1.52D	∞	∞	1.30D	∞	∞	∞	∞	∞	1.39C	∞	∞	∞	∞
N					1.52D	∞	∞		∞	∞	∞	∞	∞	1.39C	∞	∞	∞	2.47N
E					1.52D	1.86E	∞		1.69E	∞	∞	∞	∞		∞	∞	∞	2.47N
I						1.86E	∞		1.69E	2.06I	∞	∞	∞		∞	∞	∞	2.47N
F						1.86E	2.09F			2.01F	∞	∞	∞		∞	∞	∞	2.47N
J							2.09F			2.01F	0.28J	2.14J	∞		∞	∞	∞	2.47N
G							2.09F				0.27G	2.14J	2.44L		2.37L	∞	∞	2.47N

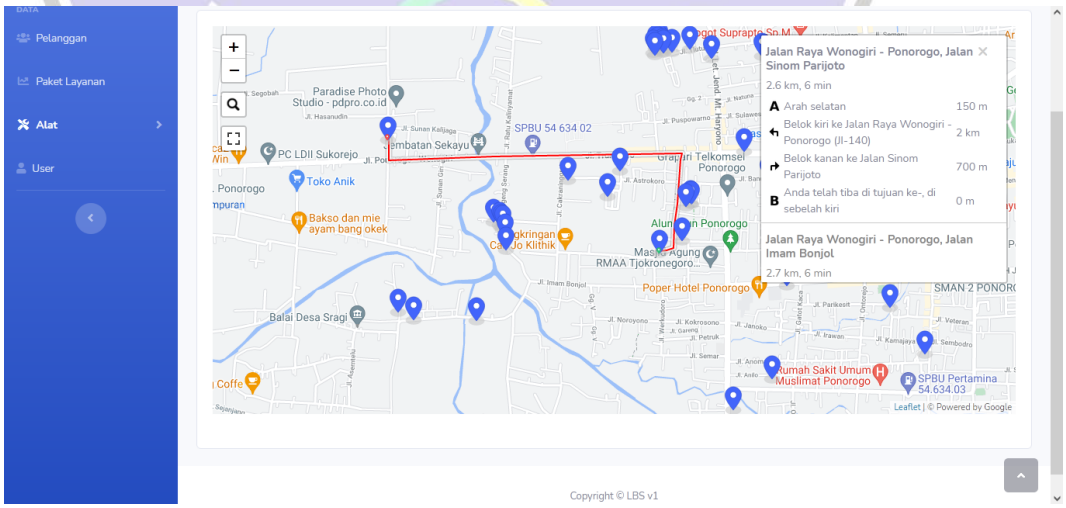
L	0.27G	2.14J	2.44L	2.37L	∞	∞	2.47N
K	0.27G		0.40K	2.37L	∞	∞	2.47N
O			0.40K	2.37L	2.69O	2.49O	2.47N
M			0.40K		0.62M	2.49O	2.47N
R					0.62M	2.49O	2.47N
Q					0.62M	2.49O	

Tabel 4. 2 Perhitungan Manual Pengujian Algoritma Pertama

Dari hasil perhitungan menggunakan algoritma dijkstra yang dilakukan secara manual, maka di dapatkan hasil rute terpendek yaitu : A-B-C-D-E-F-J-L-O-Q, sehingga akan menghasilkan graf seperti pada gambar dibawah ini.



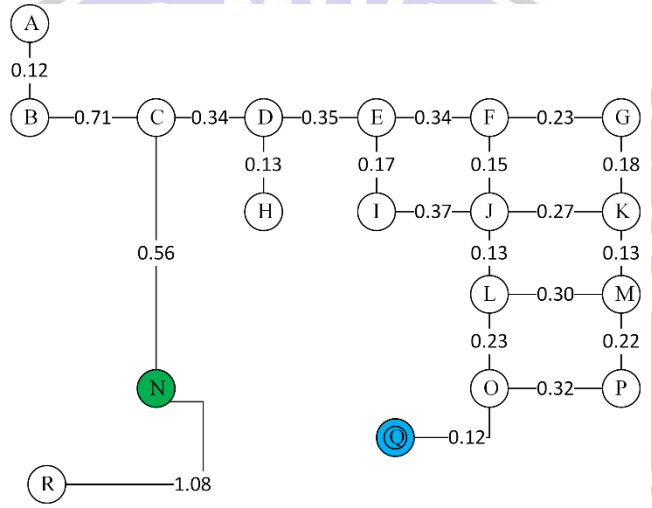
Gambar 4. 18 Hasil Graf Perhitungan Manual Pertama



Gambar 4. 19 Hasil Pengujian Algoritma Dijkstra Pada Sistem Pertama

Dari pengujian pertama penerapan algoritma dijkstra dalam sistem, sudah berhasil serta mampu memberikan rekomendasi rute terdekat yang harus dilalui agar sampai ke lokasi pelanggan tersebut dan sudah sesuai dengan perhitungan yang dilakukan secara manual.

Pengujian kedua dilanjutkan dengan mencari rute dari lokasi Bapak Sudarsono yang beralamatkan Jl. Julajuli menuju ke lokasi pelanggan yang bernama Bapak Lukman yang beralamatkan di Tepeng, Pinggirsari. Menggunakan perhitungan algoritma dijkstra secara manual.



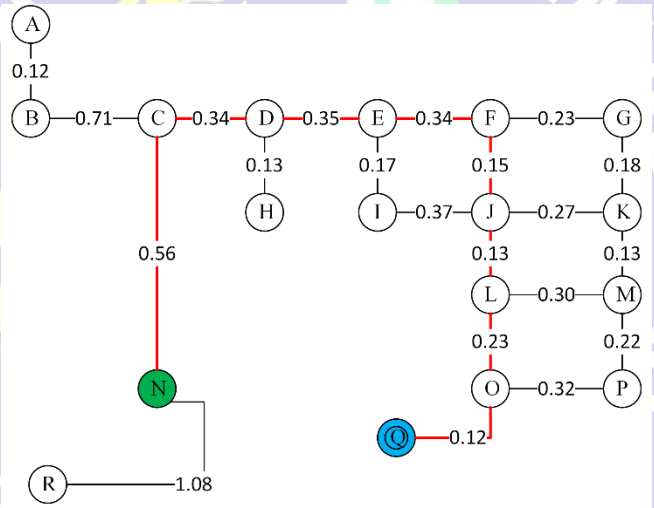
Gambar 4. 20 Permodelan Graf Pengujian Kedua

V	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Q	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.12Q	∞	0Q	∞
O	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.35O	∞	∞	0.12Q	0.32O		∞
P	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.35O	0.54P	∞		0.32O		∞
L	∞	∞	∞	∞	∞	∞	∞	∞	∞	0.48L	∞	0.35O	0.54P	∞				∞
J	∞	∞	∞	∞	∞	0.63J	∞	∞	0.85J	0.48L	0.75J		0.54P	∞				∞
M	∞	∞	∞	∞	∞	0.63J	∞	∞	0.85J		0.67M		0.54P	∞				∞
F	∞	∞	∞	∞	0.97F	0.63J	0.86F	∞	0.85J		0.67M			∞				∞
K	∞	∞	∞	∞	0.97F		0.85K	∞	0.85J		0.67M			∞				∞
G	∞	∞	∞	∞	0.97F		0.85K	∞	0.85J					∞				∞
I	∞	∞	∞	∞	0.97F			∞	0.85J					∞				∞

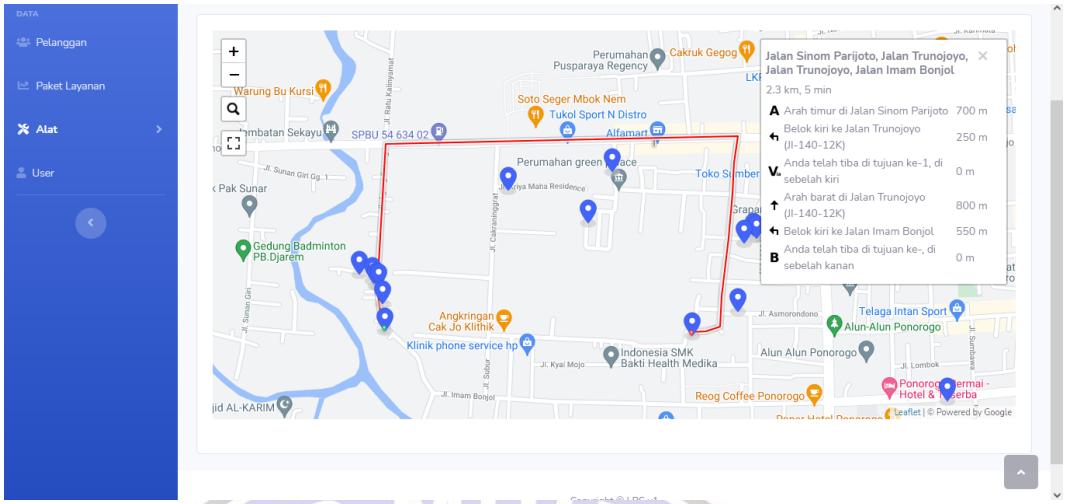
E	∞	∞	∞	1.32E	0.97F	∞	∞	∞
D	∞	∞	1.66D	1.32E		1.45D	∞	∞
H	∞	∞	1.66D			1.45D	∞	∞
C	∞	2.37C	1.66D				2.22C	∞
N	∞	2.37C					2.22C	3.30N
B	2.49B	2.37C						3.30N
R	2.49B							3.30N

Tabel 4. 3 Perhitungan Manual Pengujian Algoritma Kedua

Dari hasil perhitungan menggunakan algoritma dijkstra yang dilakukan secara manual, maka di dapatkan hasil rute terpendek yaitu : Q-O-L-J-F-E-D-C-N-R, sehingga akan menghasilkan graf seperti pada gambar dibawah ini.



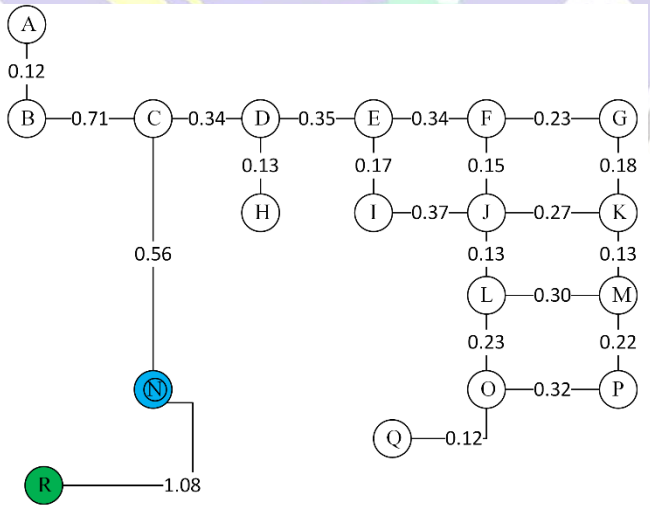
Gambar 4. 21 Hasil Graf Perhitungan Manual Kedua



Gambar 4. 22 Hasil Pengujian Algoritma Dijkstra Pada Sistem Kedua

Dari pengujian kedua penerapan algoritma dijkstra dalam sistem, juga berhasil serta mampu memberikan rekomendasi rute terdekat yang harus dilalui agar sampai ke lokasi pelanggan tersebut dan sudah sesuai dengan perhitungan yang dilakukan secara manual.

Pengujian ketiga dilanjutkan dengan mencari rute dari lokasi Bapak Lukman yang beralamatkan di Tepeng, Pinggirsari menuju ke lokasi pelanggan yang bernama Bapak Daroh yang beralamatkan di Dukuh Mancaan, Paju. Menggunakan perhitungan algoritma dijkstra secara manual.

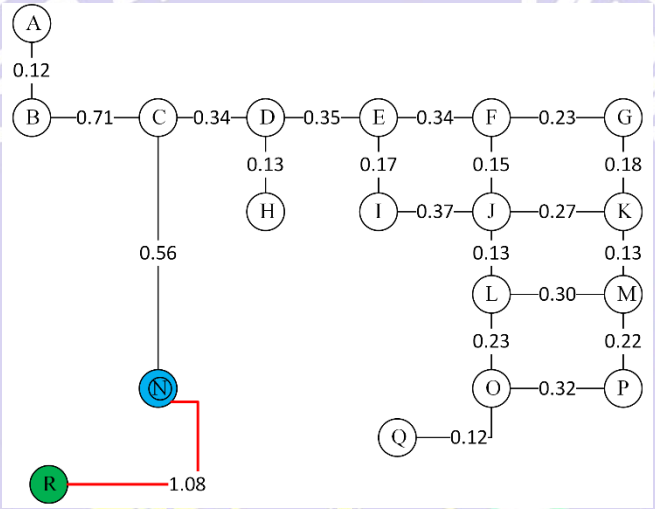


Gambar 4. 23 Permodelan Graf Pengujian Ketiga

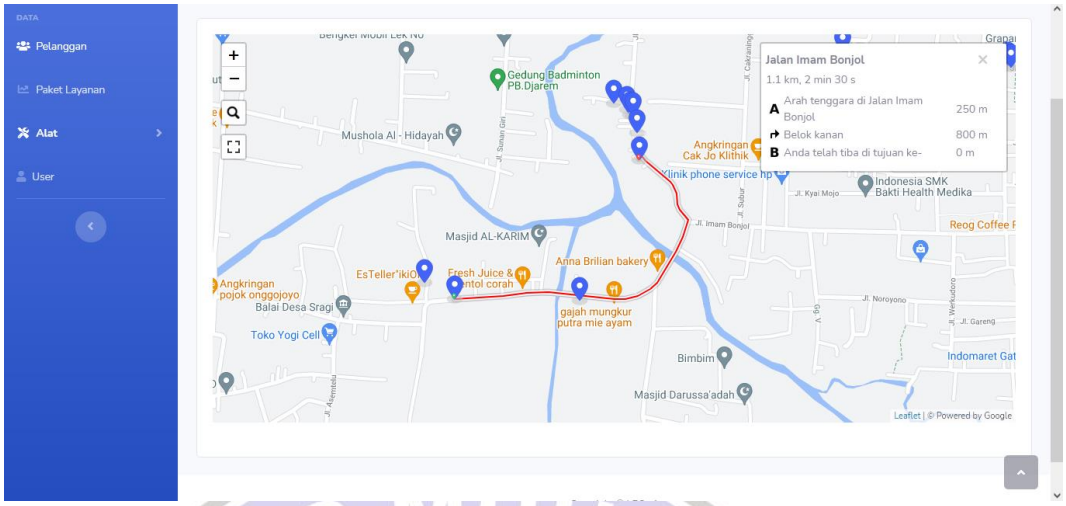
V	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
N	∞	∞	0.56N	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	0N	∞	∞	∞	1.08N
C	∞	1.27C	0.56N	0.90C	∞	∞	∞	∞	∞	∞	∞	∞	∞		∞	∞	∞	1.08N
D	∞	1.27C		0.90C	1.25D	∞	∞	1.03D	∞	∞	∞	∞	∞		∞	∞	∞	1.08N
H	∞	1.27C			1.25D	∞	∞	1.03D	∞	∞	∞	∞	∞		∞	∞	∞	1.08N
R	∞	1.27C			1.25D	∞	∞		∞	∞	∞	∞	∞		∞	∞	∞	1.08N

Tabel 4. 4 Perhitungan Manual Pengujian Algoritma Ketiga

Dari hasil perhitungan menggunakan algoritma dijkstra yang dilakukan secara manual, maka di dapatkan hasil rute terpendek yaitu : N-R, sehingga akan menghasilkan graf seperti pada gambar dibawah ini.



Gambar 4. 24 Hasil Graf Perhitungan Manual Ketiga



Gambar 4. 25 Hasil Pengujian Algoritma Dijkstra Pada Sistem Ketiga

Dari pengujian ketiga penerapan algoritma dijkstra dalam sistem, tetap berhasil serta mampu memberikan rekomendasi rute terdekat yang harus dilalui agar sampai ke lokasi pelanggan tersebut dan sudah sesuai dengan perhitungan yang dilakukan secara manual.

4.3.3 Hasil Pengujian User

Berikut ini merupakan hasil pengujian oleh user yang dilakukan di Asfimediagroup. Hasil pengujian akan ditampilkan pada tabel dibawah ini :

No	Pertanyaan	Ya	Cukup	Tidak
1	Apakah menu dan fitur aplikasi mudah digunakan ?		√	
2	Apakah aplikasi memiliki fungsi yang diharapkan ?	√		
3	Apakah aplikasi sesuai kebutuhan ?	√		
4	Apakah aplikasi mempermudah pengelolaan data?		√	
5	Apakah aplikasi memberikan rekomendasi rute terdekat menuju lokasi pelanggan?	√		

Tabel 4. 5 Hasil Pengujian User Pertama

No	Pertanyaan	Ya	Cukup	Tidak
1	Apakah menu dan fitur aplikasi mudah digunakan ?	√		
2	Apakah aplikasi memiliki fungsi yang diharapkan ?		√	
3	Apakah aplikasi sesuai kebutuhan ?	√		
4	Apakah aplikasi mempermudah pengelolaan data?	√		
5	Apakah aplikasi memberikan rekomendasi rute terdekat menuju lokasi pelanggan?	√		

Tabel 4. 6 Hasil Pengujian User Kedua

No	Pertanyaan	Ya	Cukup	Tidak
1	Apakah menu dan fitur aplikasi mudah digunakan ?	√		
2	Apakah aplikasi memiliki fungsi yang diharapkan ?	√		
3	Apakah aplikasi sesuai kebutuhan ?	√		
4	Apakah aplikasi mempermudah pengelolaan data?	√		
5	Apakah aplikasi memberikan rekomendasi rute terdekat menuju lokasi pelanggan?		√	

Tabel 4. 7 Hasil Pengujian User Ketiga