BAB V
IMPLEMENTASI

Pada bab ini dibahas mengenai implementasi perangkat lunak berdasarkan hasil yang telah diperoleh dari analisis kebutuhan dan proses perancangan perangkat lunak yang dibuat. Pembahasan terdiri dari penjelasan tentang spesifikasi sistem, batasan-batasan dalam implementasi, implementasi algoritma pada sistem, dan implementasi antarmuka beserta fungsi proseduralnya.

5.1 Spesifikasi Sistem

Hasil analisis kebutuhan dan perancangan perangkat lunak yang telah diuraikan pada Bab IV menjadi acuan untuk melakukan implementasi menjadi sistem yang dapat berfungsi sesuai dengan kebutuhannya. Spesifikasi sistem diimplementasikan pada spesifikasi perangkat keras dan perangkat lunak.

5.1.1 Spesifikasi Perangkat Keras

Pembangunan sistem pendukung keputusan pemilihan transportasi umum Kota Malang ini menggunakan spesifikasi perangkat keras yang terdapat pada pada **Tabel 5.1**.

Tabel 5.1 Spesifikasi Perangkat Keras Komputer

Nama Komponen	Spesifikasi
Prosesor	Intel® Pentium® Dual-Core CPU processor T4200 (2.0 GHz, 800 MHz, 1MB L2 cache)
Memori(RAM)	1978MB DDR3
Hardisk	250 GB HDD
Kartu Grafis	Mobile Intel® 4 Series Express Chipset Family 256 MB
Monitor	14.0” HD Acer CineCrystal™ LED LCD

Sumber: Implementasi

5.1.2 Spesifikasi Perangkat Lunak

Pembangunan sistem pendukung keputusan pemilihan transportasi umum Kota Malang ini menggunakan perangkat lunak dengan spesifikasi yang dijelaskan pada **Tabel 5.2**.

Tabel 5.2 Spesifikasi Perangkat Lunak Komputer

Nama Komponen	Spesifikasi
Sistem Operasi	Microsoft Windows XP Professional (5.1, Build 2600)
Bahasa Pemrograman	PHP
Tools pemrograman	Adobe Dreamweaver CS3
Web Server	Apache 2.2.17
Web Browser	Mozilla Firefox
DBMS	MySQL

Sumber: Implementasi

5.2 Batasan-Batasan Implementasi

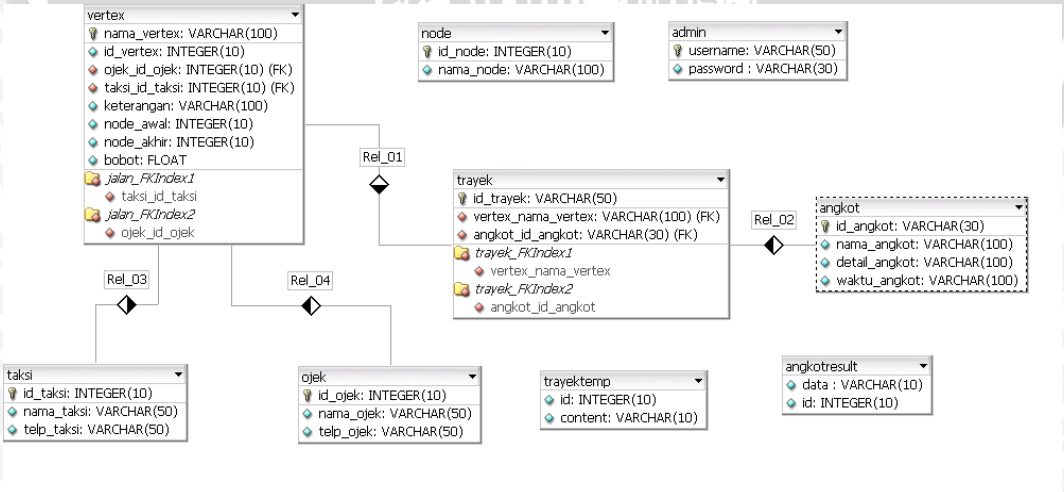
Beberapa batasan dalam mengimplementasikan sistem adalah sebagai berikut:

- Input yang diterima oleh sistem adalah lokasi awal dan lokasi tujuan yang berupa nama jalan atau nama bangunan dan tempat penting yang oleh sistem dianggap sebagai vertex serta prioritas dasar keputusan *user*.
- Output yang diterima oleh *user* adalah rute yang dilewati berupa nama jalan tanpa peta, jarak terpendek, alat transportasi yang dapat digunakan, estimasi waktu, tarif, dan rekomendasi.
- Algoritma yang digunakan dalam penentuan jalur terpendek adalah *Floyd-Warshall*.
- Obyek yang diproses dalam perhitungan jarak terpendek hanya *node* yang saling berhubungan berupa vertex atau jalan yang datanya ada di dalam database.
- Jalan yang menjadi obyek adalah jalan utama atau umum di Kota Malang yang dapat dilewati angkot dan bukan jalan kecil atau gang.
- Parameter yang digunakan dalam menentukan lintasan terpendek hanya bergantung pada jarak tempuh dan diasumsikan kondisi lalu lintas normal lancar, lampu lalu lintas diabaikan.
- Satu jenis transportasi diasumsikan memiliki kecepatan sama.

- Tarif yang dihasilkan oleh sistem adalah tarif sesuai rumus, kegiatan tawar menawar antara pengemudi dan penumpang diabaikan.
- Apabila terdapat suatu kasus dimana suatu jalan tidak dilalui angkot satu pun dan total panjang jalan kurang dari 500 meter, maka *user* diberi saran untuk berjalan kaki.
- Rekomendasi angkot hanya akan tampil jika terjadi oper kurang dari tiga kali.
- Antarmuka sistem dapat digunakan secara langsung oleh *user*, tetapi diharuskan melakukan proses login terlebih dahulu bagi *administrator*.
- *User* sebelumnya telah mengetahui nama jalan atau tempat yang menjadi lokasi awal dan tujuan.
- Eksekusi program berjalan sesuai dengan waktu eksekusi algoritma *Floyd-Warshall*, yaitu $\Theta(v^3)$.
- Versi XAMPP yang digunakan adalah versi 1.7.4.
- *Web browser* yang digunakan selama proses implementasi adalah Mozilla Firefox.
- *Database* sistem atau basis pengetahuan disimpan dalam DBMS MySQL.

5.3.1 Implementasi Penyimpanan Data

Implementasi penyimpanan data dilakukan dengan *Database Management System* MySQL. Hasil implementasi SQL pada *database* ini dimodelkan dalam diagram konseptual *entity relationship* seperti pada **Gambar 5.1**.



Gambar 5.1 *Physical Diagram* SPK Pemilihan Alat Transportasi Umum
Sumber: Implementasi

Relasi:

1. Relasi Rel_01, relasi ini untuk menghubungkan tabel jalan dengan tabel trayek, dimana hubungan yang terjadi adalah 1 ke n. Satu jalan dapat dimiliki oleh banyak trayek angkot di tabel trayek.
2. Relasi Rel_02, relasi ini untuk menghubungkan tabel angkot dengan tabel trayek, dimana hubungan yang terjadi adalah 1 ke n. Satu jenis angkot dapat dimiliki oleh banyak trayek angkot di tabel trayek.
3. Relasi Rel_03, relasi ini untuk menghubungkan tabel taksi dengan tabel jalan, dimana hubungan yang terjadi adalah 1 ke n. Satu perusahaan taksi dapat dimiliki oleh banyak jalan di tabel jalan. Hal ini digunakan untuk kepentingan bisnis.
4. Relasi Rel_04, relasi ini untuk menghubungkan tabel ojek dengan tabel jalan, dimana hubungan yang terjadi adalah 1 ke n. Satu nama pemilik ojek dapat dimiliki oleh banyak jalan di tabel jalan. Hal ini digunakan untuk kepentingan bisnis.

Berikut, merupakan struktur tabel dalam *physical diagram*:

Tabel Vertex

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id_vertex	Int	10		NN
2	nama_vertex	Varchar	100	PK	NN
3	keterangan	Varchar	100		
4	node_awal	Int	10		NN
5	node_akhir	Int	10		NN
6	Bobot	Float	-		NN
7	id_taksi	Int	10	FK	
8	id_ojek	Int	10	FK	

Tabel Node

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id_node	Int	10	PK	NN
2	nama_node	Varchar	100		

Tabel Trayek

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id_trayek	Varchar	50	PK	NN
4	nama_vertex	varchar	100	FK	NN
5	id_angkot	varchar	30	FK	NN

Tabel Angkot

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id_angkot	Varchar	30	PK	NN
2	nama_angkot	Varchar	100		
3	detail_angkot	Varchar	100		
4	waktu_angkot	Varchar	100		

Tabel Taksi

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id_taksi	Int	10	PK	NN
2	nama_taksi	varchar	50		
3	telp_taksi	varchar	50		

Tabel Ojek

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id_ojek	Int	10	PK	NN
2	nama_ojek	varchar	50		
3	telp_ojek	varchar	50		

Tabel Admin

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	username	varchar	50	PK	NN
2	password	varchar	30		

Tabel Trayektemp

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	id	integer	10		
2	content	varchar	10		

Tabel Angkotresult

No.	Nama Field	Tipe	Ukuran	PK/FK	NN
1	data	varchar	10		
2	id	integer	10		

5.4 Implementasi Algoritma *Floyd-Warshall*

Proses pencarian rute terpendek dilakukan dengan menggunakan algoritma *Floyd-Warshall*. Algoritma ini merupakan jenis algoritma dinamis yang akan memproses semua pasangan titik yang ada. Algoritma ini menggunakan banyak perulangan untuk memproses semua titik sampai habis. Setelah diketahui semua jarak dari pasangan titik, maka sistem tinggal memilih rute yang pasangan titiknya sesuai dengan yang diinputkan oleh *user*.

```
1.  //initialization
2.  for ( $i = 1; $i <= $nodess; $i++ ) {
3.  for ( $j = 1; $j <= $nodess; $j++ ) {
4.  if (empty($weights[$i][$j]))
5.  $weights[$i][$j]=0;
6.  if ( $i == $j ) {
7.  $dist[$i][$j] = 0;
8.  } else if ( $weights[$i][$j] > 0) {
9.  $dist[$i][$j] = $weights[$i][$j];
10. } else {
11. $dist[$i][$j] = pow(2, (20 * 8 - 2)-1);}
12. $pred[$i][$j] = $i; } }
13.
14. //algoritma floyd_warshall
15. for ( $k = 1; $k <= $nodess; $k++ ) {
16. for ( $i = 1; $i <= $nodess; $i++ ) {
17. for ( $j = 1; $j <= $nodess; $j++ ) {
18. if ($dist[$i][$j] > ($dist[$i][$k] + $dist[$k][$j])) {
19. $dist[$i][$j] = $dist[$i][$k] + $dist[$k][$j];
20. $pred[$i][$j] = $pred[$k][$j]; } } } }
```

Gambar 5.2 Implementasi Algoritma *Floyd-Warshall* Dalam Proses Pencarian Rute Terpendek
Sumber: Implementasi

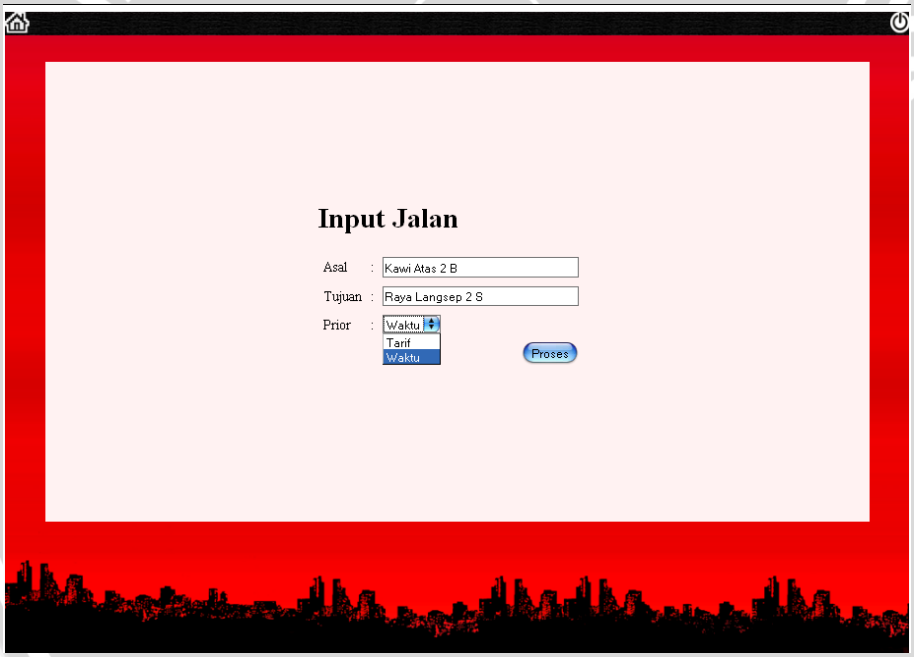
Penjelasan implementasi algoritma *Floyd-Warshall* untuk mencari rute terpendek pada **Gambar 5.2** yaitu:

1. Variabel i merupakan titik awal, j merupakan titik akhir, k merupakan titik yang dilewati (antara i dan j), nodess merupakan jumlah dari semua titik yang diketahui yang ada di dalam database, dist merupakan bobot antar titik, dan pred untuk menyimpan titik yang dilewati.
2. Baris 2-12 adalah tahap initialization atau permulaan yang merupakan pengetahuan awal yang mengadopsi data yang hanya diketahui di dalam database.
3. Baris 2-3 merupakan perulangan untuk persiapan penghitungan setiap titik.
4. Baris 4-11 dilakukan pengisian bobot antar titik sesuai dengan database sebagai awal yang diketahui. Awalnya bobot antara i dan j diisi 0 semua. Jika titik i=j maka bobotnya adalah 0 karena sama. Jika jarak i ke j lebih dari 0, maka bobotnya diisi jarak tersebut. Jika titik i dan j tidak berhubungan maka diisi tak terhingga yang oleh penulis menggunakan rumus pow(2, (20 * 8 - 2)-1).
5. Baris 12 menunjukkan bahwa sebagai awal, antara titik i ke j melewati titik i karena pada tahap ini hanya memproses data yang ada di dalam database, yaitu data titik yang secara langsung berhubungan.
6. Baris 15-20 merupakan tahap inti algoritma *Floyd-Warshall*.
7. Baris 15-17 merupakan perulangan untuk penghitungan setiap titik.
8. Baris 18-19 merupakan proses perhitungan bobot. Jika bobot i ke j lebih besar daripada total bobot antara i ke k dan k ke j maka jarak i ke j diganti dengan total bobot antara i ke k dan k ke j. Ini dilakukan untuk mencari jarak yang terpendek dan berhubungan.
9. Baris 20 menunjukkan bahwa jika proses pada baris 18 terpenuhi, maka titik yang dilalui dari i ke j diisi dengan titik j ke k.
10. Proses 15-20 dilakukan berulang terus menerus dengan membandingkan semua titik sampai habis.

5.5 Implementasi Antarmuka

5.5.1 Implementasi Halaman Utama *Website*

Pada halaman utama *website, user* dihadapkan pada dua buah *text box* yang dapat digunakan *user* untuk memberikan inputan nama jalan atau nama tempat dan prioritas pengambilan keputusan. *Text box* yang pertama diisi asal sedangkan yang kedua diisi lokasi yang menjadi tujuan *user*. *Combo box* menyediakan pilihan prioritas pengambilan keputusan, berdasar tarif atau waktu. Setelah semua box terisi, *user* dapat melihat hasil dari proses pendukung keputusan dengan menekan tombol 'proses'. Tampilan halaman utama diperlihatkan pada **Gambar 5.3**.



Gambar 5.3 Tampilan Halaman Utama *Website*
Sumber: Implementasi

5.5.2 Implementasi Halaman Hasil Pendukung Keputusan

Pada halaman hasil rute terpendek, *user* dihadapkan pada tampilan jarak terpendek, rute yang akan dilalui, dan transportasi umum terbaik berdasarkan waktu tercepat atau tarif termurah sesuai dengan pilihan *user*. Hasil tersebut telah melalui proses algoritma *Floyd-Warshall*.

Ada beberapa tahap dalam memproses data sehingga menghasilkan sebuah pendukung keputusan. Tahap tersebut antara lain:

a. Mengambil Data Inputan

Data inputan dari *user* yang berupa jalan atau vertex diproses dengan memanggil node dari database. Berikut merupakan cuplikan *script* proses pengambilan data inputan:

```
<?php
$asal = $_POST ["asal"];
$tujuan = $_POST ["tujuan"];
$prior = $_POST ["prior"];
$sqla = mysql_query("SELECT * FROM vertex WHERE nama_vertex =
'$asal'");
$sqlt = mysql_query("SELECT * FROM vertex WHERE nama_vertex =
'$tujuan'");
while ($sqlasal = mysql_fetch_row($sqla)){
$awal = $sqlasal[4];
while ($sqlakhir = mysql_fetch_row($sqlt)){
$akhir = $sqlakhir[5];

if ($asal == $tujuan) {
echo"<script>alert('Masukan Salah!')</script>";
echo"<html><head><meta http-equiv='refresh'
content='0;url=home.php'></head><body></body></html>";
}
if ($asal == "") {
echo"<script>alert('Masukan Salah!')</script>";
echo"<html><head><meta http-equiv='refresh'
content='0;url=home.php'></head><body></body></html>";
}
if ($tujuan == "") {
echo"<script>alert('Masukan Salah!')</script>";
echo"<html><head><meta http-equiv='refresh'
content='0;url=home.php'></head><body></body></html>";}
else {
echo "<br/>Lokasi Asal : <strong>$asal</strong> ----> Lokasi
tujuan : <strong>$tujuan</strong></br>";
```

b. Perhitungan dengan *Floyd-Warshall*

Algoritma *Floyd-Warshall* merupakan algoritma pencarian lintasan terpendek antara semua pasangan simpul (all pairs shortest path). Seluruh data node dan vertex yang ada di dalam database akan diproses terlebih dahulu oleh algoritma ini sebelum dicocokkan dengan inputan *user*. Berikut merupakan cuplikan *script* proses perhitungan dengan *Floyd-Warshall*:

```
//persiapan
//ambil data di tabel node, ambil jumlahnya
$sqlnode = mysql_query("SELECT * FROM node ORDER BY id_node ASC");
$jmlnode = mysql_num_rows($sqlnode);

//membuat array kosongan untuk persiapan perhitungan
(barisxkolom), jumlah tergantung dari jumlah node (nodexnode)
$graph = array();
for($i = 1; $i < $jmlnode; $i++){
for ($m = 1; $m < $jmlnode; $m++) {
$graph[$i][$m] = "0"; //isi bobot, dikasih nilai awal 0 semua
}
}

//ambil data dari tabel vertex, kolom node_awal, node_akhir, bobot
$sqlvertex = mysql_query("SELECT * FROM vertex ORDER BY node_awal
ASC");
//mengambil bobot antar node yang diketahui dari database
//mengisi array kosongan dg bobot yg diketahui
while ($queryv = mysql_fetch_array($sqlvertex)) {
$gasal = $queryv[4];
$gtujuan = $queryv[5];
$graph[$gasal][$gtujuan] = $queryv[6];
}

//masukkan node dalam array sesuai dengan awal-akhir di database
$nodes = array();
//$titiknodes = array();
$i = 0;
while ($data = mysql_fetch_array($sqlnode)) {
$nodes[$i] = $data[0];
$i++;
}

$weights = $graph;
$nodess = count($weights);

//Algoritma
//initialization
for ( $i = 1; $i <= $nodess; $i++ ) {
for ( $j = 1; $j <= $nodess; $j++ ) {
if (empty($weights[$i][$j]))
$weights[$i][$j]=0;
if ( $i == $j ) {
$dist[$i][$j] = 0;
} else if ( $weights[$i][$j] > 0) {
$dist[$i][$j] = $weights[$i][$j];
} else {
$dist[$i][$j] = pow(2, (20 * 8 - 2)-1);}
```

```
$pred[$i][$j] = $i; } }

//algoritma floyd_warshall
for ( $k = 1; $k <= $nodess; $k++ ) {
for ( $i = 1; $i <= $nodess; $i++ ) {
for ( $j = 1; $j <= $nodess; $j++ ) {
if ($dist[$i][$j] > ($dist[$i][$k] + $dist[$k][$j])) {
$dist[$i][$j] = $dist[$i][$k] + $dist[$k][$j];
$pred[$i][$j] = $pred[$k][$j]; } } } }
```

c. Pengambilan Data Jarak

Setelah data diproses oleh algoritma *Floyd-Warshall* maka dilakukan pengambilan data bobot sesuai dengan inputan asal dan tujuan *user*. Berikut merupakan cuplikan *script* proses pengambilan data jarak terpendek:

```
//perhitungan jarak
function get__distance($i, $j) {
global $dist;
return $dist[$i][$j];
}
//print distance
$dt = get__distance($awal,$akhir);
echo "Jarak Terpendek: <strong> $dt meter </strong>";
echo "</br>";
```

d. Pengambilan Data Rute

Setelah didapatkan data hasil proses algoritma *Floyd-Warshall*, maka dapat ditelusuri rute yang akan dilalui. Berikut merupakan cuplikan *script* proses pengambilan data rute:

```
$tmp= array(); //temporary untuk bantuan mengisi jarak
//untuk proses pengambilan pair path, dinamis
function getpath($i, $j) {
global $pred;
global $tmp;
if ( $i != $j ) {
getpath($i, $pred[$i][$j]); //kembali fungsi ke atas sehingga j--
>predesesor, satu step di belakangnya
}
array_push($tmp, $j);
//echo $j ;
}

//pemanggilan fungsi disesuaikan dg inputan
function get__path($i, $j) {
global $tmp;
getpath($i, $j);
```

```
return $tmp;
}
$rute = get__path($awal,$akhir);

//cari jalan road
//pengambilan nama jalan, bobot
$ctmp=count ($tmp); //jumlah node yang dilewati
$jml=count ($tmp); //untuk angkot biar tidak berubah
$tmp1="";
$tmpjln="";
echo "Rute:";
echo "</br>";
$y=0;
for ($r=0; $r < $ctmp-1; $r++) //r= road, penunjuk array dalam tmp
| ctmp-1 karena jalannya = jumlah node-1
{
$tmpa= $tmp[$r]; //node awal
$tmpb= $tmp[$r+1]; //node akhir
$sqlroad= mysql_query ("SELECT * FROM vertex WHERE node_awal=$tmpa
AND node_akhir=$tmpb"); //query buat ambil nama jalan+bobot
$sqlr= mysql_fetch_array ($sqlroad);
//ambil global jalan
if ($r==0)
{
$tmpjln=$sqlr[2];
echo "-- Jalan $tmpjln</br>";
}
elseif($r!=0 && $sqlr[2]==$tmpjln)
{
$tmpjln=$sqlr[2];
}
elseif($r!=0 && $sqlr[2]!=$tmpjln)
{
$tmpjln=$sqlr[2];
echo "-- Jalan $tmpjln</br>";
}
```

e. Perhitungan Tarif dan Waktu Transportasi Umum

Setelah diketahui data jarak, maka dapat diproses perhitungan estimasi waktu dan tarif masing-masing transportasi umum. Berikut merupakan cuplikan *script* proses perhitungan tarif dan waktu:

```
//perhitungan angkot tarif waktu
if($noter>0)
{
$wangkot = 10000000000;
$wangkotfix = 10000000000;
$tangkot = 10000000000;
```

```
$tangkotfix = 10000000000;
}
elseif($y>3)
{
$wangkot = 10000000000;
$wangkotfix = 10000000000;
$tangkot = 10000000000;
$tangkotfix = 10000000000;
}
else
if($kaki>0)
{
//tarif
$tangkot = ($y*2500);
$tangkotfix = number_format($tangkot);
//waktu
$wangkot = ($dt/300) + (($y-1)*3) + $wkaki;
$wangkotfix = sprintf ("%01.2f", $wangkot);
}
else
{
//tarif
$tangkot = ($y*2500);
$tangkotfix = number_format($tangkot);
//waktu
$wangkot = ($dt/300) + (($y-1)*3);
$wangkotfix = sprintf ("%01.2f", $wangkot);
}}

//perhitungan ojek
$wojek = ($dt/600);
$wojekfix = sprintf ("%01.2f", $wojek);
$tojek = (($dt/1000)*2000);
$tojekfix = number_format ($tojek);

//perhitungan taksi
$wtaksi = ($dt/550);
$wtaksifix = sprintf ("%01.2f", $wtaksi);
$ttaksi1 = (($dt/1000)*3000) + 6000;
$ttaksi2 = $ttaksi1;
if ($ttaksi2>20000)
{
$ttaksifix = number_format ($ttaksi1);
//echo "Rp. $ttaksifix1,00 <br/>";
}
else
{
$ttaksi2 = 20000;
```

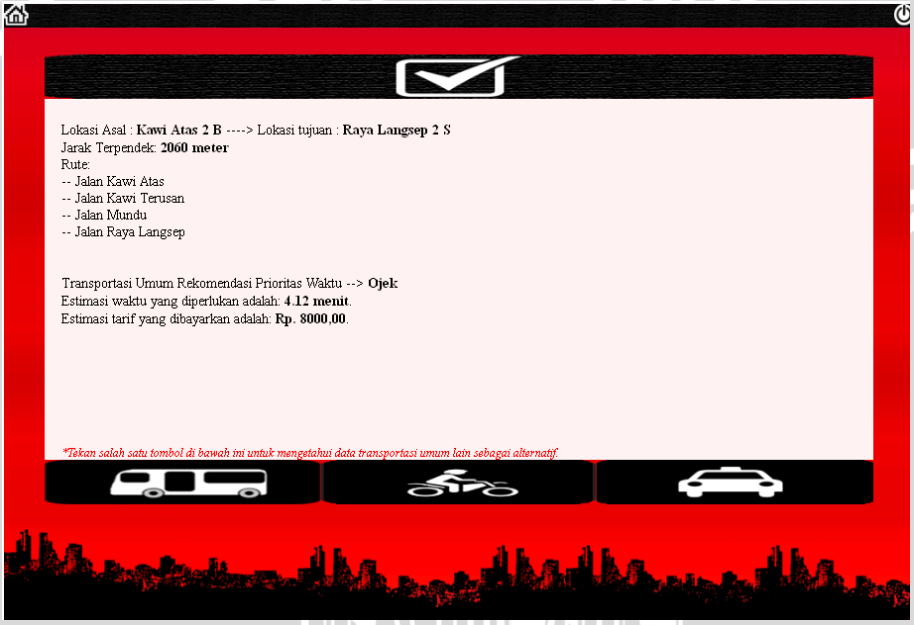
```
//echo "          *karena harga minimal taksi adalah Rp 20.000,00
maka anda harus membayar harga minimal";
}
```

f. Penentuan Transportasi Rekomendasi Berdasar Prioritas

Dari hasil estimasi tarif dan waktu masing-masing transportasi umum, dapat ditampilkan pendukung keputusan sesuai dengan prioritas *user*. Berikut merupakan cuplikan *script* proses pengambilan keputusan berdasar prioritas:

```
//tampil data sesuai prioritas
if ($prior=="tarif"){
echo "<br/><br/>Transportasi Umum Rekomendasi Prioritas Tarif: ";
if (($tangkot<=$tojek) && ($tangkot<=$ttaksi2)){
echo "<strong> Angkot </strong> <br/> ";
echo "Estimasi tarif yang dibayarkan adalah: Rp. $tangkotfix,00
<br/>";
echo "Estimasi waktu yang diperlukan adalah: $wangkotfix menit
<br/>";}

elseif (($tojek<=$tangkot) && ($tojek<=$ttaksi2)){
echo "<strong> Ojek </strong> <br/> ";
echo "Estimasi tarif yang dibayarkan adalah: Rp. $tojek,00 <br/>";
echo "Estimasi waktu yang diperlukan adalah: $wojekfix menit
<br/>";}

elseif (($ttaksi2<=$tangkot) && ($ttaksi2<=$tojek)){
echo "<strong> Taksi </strong> <br/> ";
echo "Estimasi tarif yang dibayarkan adalah: Rp. $ttaksifix,00
<br/>";
if ($ttaksifix <20000) {
echo "      *karena harga minimal taksi adalah Rp 20.000,00 maka
anda harus membayar harga minimal";}
echo "Estimasi waktu yang diperlukan adalah: $wtaksifix menit
<br/>";}}
else{
echo "<br/><br/>Transportasi Umum Rekomendasi Prioritas Waktu: ";
if (($wangkot<=$wojek) && ($wangkot<=$wtaksi)){
echo "<strong> Angkot </strong> <br/> ";
echo "Estimasi waktu yang diperlukan adalah: $wangkotfix menit
<br/>";
echo "Estimasi tarif yang dibayarkan adalah: Rp. $tangkotfix,00
<br/>";}

elseif (($wojek<=$wangkot) && ($wojek<=$wtaksi)){
echo "<strong> Ojek </strong> <br/> ";
echo "Estimasi waktu yang diperlukan adalah: $wojekfix menit
<br/>";
```

```
echo "Estimasi tarif yang dibayarkan adalah: Rp. $tojekfix,00
<br/>";}

elseif (($wtaksi<=$wangkot) && ($wtaksi<=$wojek)){
echo "<strong> Taksi </strong> <br/> ";
echo "Estimasi waktu yang diperlukan adalah: $wtaksifix menit
<br/>";
echo "Estimasi tarif yang dibayarkan adalah: Rp. $ttaksifix,00
<br/>";
if ($ttaksifix <20000) {
echo "      *karena harga minimal taksi adalah Rp 20.000,00 maka
anda harus membayar harga minimal";}}}
```



Gambar 5.4 Tampilan Halaman Hasil Pendukung Keputusan
Sumber: Implementasi

5.5.3 Implementasi Halaman Detail Transportasi Angkot

Pada halaman detail transportasi umum angkutan kota ditampilkan hasil trayek angkutan kota yang dapat dilewati jika menggunakan rute hasil algoritma *Floyd-Warshall*. Pada halaman ini juga akan ditampilkan estimasi waktu, tarif yang akan dibayarkan, dan angkutan kota alternatif jika ada. Angkutan kota alternatif ditampilkan jika ada satu jenis angkutan kota yang melewati lokasi asal dan tujuan *user* dengan tanpa memperhatikan perhitungan dengan algoritma. Halaman ini dibuat sebagai alternatif pilihan jika hasil perhitungan sistem pendukung keputusan tidak mengarah pada transportasi umum angkutan kota. Berikut merupakan cuplikan *script* proses pengambilan data trayek angkutan kota.

Store Procedure pengambilan angkot terbaik:

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `spktrans`.`cariangkotfinal`$$
CREATE DEFINER=`root`@`localhost` PROCEDURE `cariangkotfinal`()
BEGIN
    DECLARE a INT;
    DECLARE b INT;
    DECLARE maks INT;
    SET b=0;
    DELETE FROM angkotresult;
    SET maks=(SELECT MAX(id) FROM trayektemp);

    REPEAT /*-5ke0*/
IF EXISTS( SELECT a.content,a.id,b.id,c.id,d.id,e.id,f.id
FROM
 (SELECT * FROM trayektemp  WHERE id=b)a
LEFT JOIN
 (SELECT * FROM trayektemp  WHERE id=b-1)b ON  a.content=b.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-2)c ON  a.content=c.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-3)d ON  a.content=d.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-4)e ON  a.content=e.content
LEFT JOIN
 (SELECT * FROM trayektemp  WHERE id=b-5)f ON  a.content=f.content
WHERE (a.id IS NOT NULL AND ((((f.id IS NOT NULL  OR e.id IS NOT
NULL) AND d.id IS NOT NULL) AND c.id IS NOT NULL) AND b.id IS NOT
NULL )))THEN

INSERT INTO angkotresult
SELECT a.content,a.id
FROM
(SELECT * FROM trayektemp  WHERE id=b)a
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-1)b ON  a.content=b.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-2)c ON  a.content=c.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-3)d ON  a.content=d.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-4)e ON  a.content=e.content
LEFT JOIN
(SELECT * FROM trayektemp  WHERE id=b-5)f ON  a.content=f.content
WHERE (a.id IS NOT NULL AND ((((f.id IS NOT NULL  OR e.id IS NOT
NULL) AND d.id IS NOT NULL) AND c.id IS NOT NULL) AND b.id IS NOT
NULL ))
            LIMIT 1;
(……)
END IF;
SET b=b+1;
    UNTIL maks-b=1 END REPEAT;
    END$$
DELIMITER ;
```

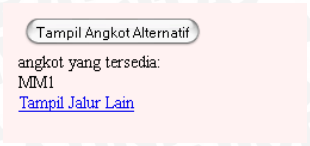
Memanggil *Stored Procedure*:

```
//ambil angkot terbaik dengan storeproc
$sqlins=mysql_query ("DELETE FROM trayektemp");
for ($r=0; $r < $ctmp-1; $r++) //r= road, penunjuk array dalam tmp
| ctmp-1 karena jalannya = jumlah node-1
{
$tmpa= $tmp[$r]; //node awal
$tmpb= $tmp[$r+1]; //node akhir
$sqlroad= mysql_query ("SELECT * FROM vertex WHERE node_awal=$tmpa
AND node_akhir=$tmpb"); //query buat ambil nama jalan+bobot
$sqlr= mysql_fetch_array ($sqlroad);
$jln= $sqlr[1];
$query="SELECT * FROM angkotview WHERE nama_vertex='$jln'";
//query angkotview ambil angkot, sesuai dengan nama jalan
$sqlangkot2= mysql_query ($query); //proses angkot temp
recommended
$z=0;
while($sqlkot2= mysql_fetch_array ($sqlangkot2)) //ngisi ke tabel
temporary
{
$has[$r][$z]=$sqlkot2[5];
$sqlins2=mysql_query ("INSERT INTO trayektemp(id, content) values
('$r', '$sqlkot2[5]')");
$z=$z+1;
}
}
$sqlproc=mysql_query ("call cariangkotfinal");
```

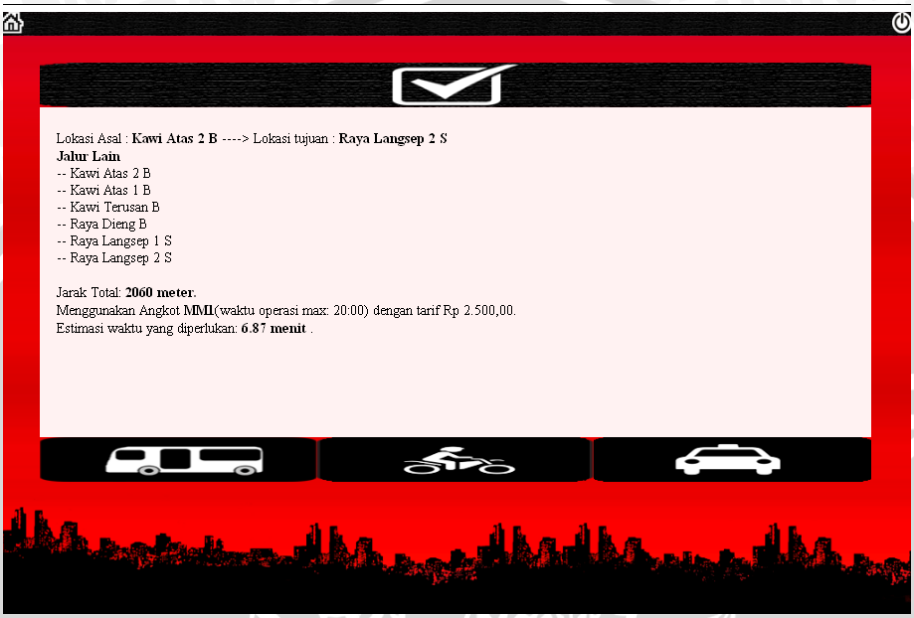
Gambar 5.5 Tampilan Halaman Detail Transportasi Umum Angkutan Kota
Sumber: Implementasi

Jika ada angkot alternatif, maka dapat pula ditampilkan jalur alternatif, tetapi tidak menggunakan algoritma *Floyd-Warshall* sehingga menghasilkan jalur yang lebih panjang. Fitur ini hanya tambahan saja dan tidak semua permasalahan menampilkan jalur alternatif. Fitur ini ditampilkan hanya jika di jalan pertama dan terakhir memiliki kode angkot yang sama. Berikut merupakan cuplikan *script* proses pengambilan jalur alternatif jika tersedia.

```
for ($jarak=0, $x=$a; $x<=$b; $x++)
{
$qryjal=mysql_query ("SELECT * FROM angkotview WHERE
id_angkot='$data' AND urutan='$x'");
while ($sqljal=mysql_fetch_array($qryjal))
{
echo "-- $sqljal[1] <br/>";
$jarak=$jarak+$sqljal[4];
}
}
```



Gambar 5.6 Tampilan Jika Tersedia Angkot dan Jalur Alternatif
Sumber: Implementasi



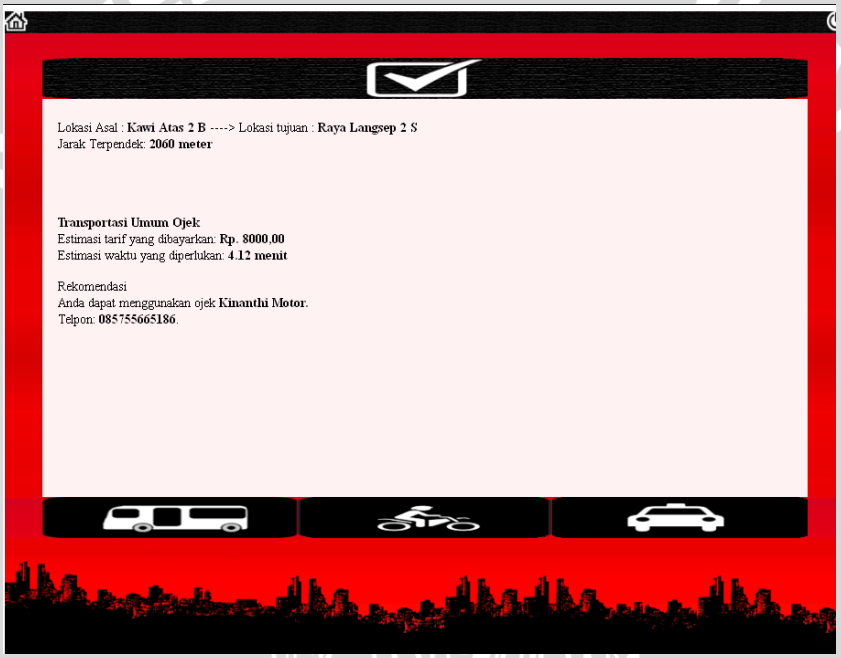
Gambar 5.7 Tampilan Halaman Angkot dan Jalur Alternatif
Sumber: Implementasi

5.5.4 Implementasi Halaman Detail Transportasi Umum Ojek

Pada halaman detail transportasi umum ojek ditampilkan hasil perhitungan estimasi waktu dan tarif yang akan dibayarkan berdasar jarak yang telah diproses dengan algoritma *Floyd-Warshall*. Pada halaman ini juga akan ditampilkan profil tukang ojek yang tersedia di jalan tersebut untuk tujuan *advertisement*. Halaman ini dibuat sebagai alternatif pilihan jika hasil perhitungan sistem pendukung keputusan tidak mengarah pada transportasi umum ojek. Berikut merupakan cuplikan *script* proses perhitungan estimasi waktu dan tarif transportasi umum ojek:

```
//tarif
echo "Estimasi tarif yang dibayarkan: ";
$dtpros = $dt-500;
if ($dtpros>1)
      {
      $dtpros= $dtpros/500;
      $dtprosceil= ceil ($dtpros);}
else
```

```
    {$dtprosceil= 0;}
$tojekfix= ($dtprosceil*1000) + 4000;
echo "<strong>Rp. $tojekfix,00 </strong> <br/>";

//waktu
echo "Estimasi waktu yang diperlukan: ";
$wojek = ($dt/500);
$wojekfix = sprintf ("%01.2f", $wojek);
echo "<strong> $wojekfix menit </strong> <br/>";

//rekomendasi
$qreko=mysql_query ("SELECT * FROM ojekview WHERE
nama_vertex='$asal'");
while ($sqlreko = mysql_fetch_row($qreko)){
echo "<br/> Rekomendasi <br/> Anda dapat menggunakan ojek <strong>
$sqlreko[2]. Telpon: <strong>$sqlreko[3]</strong>."}
```



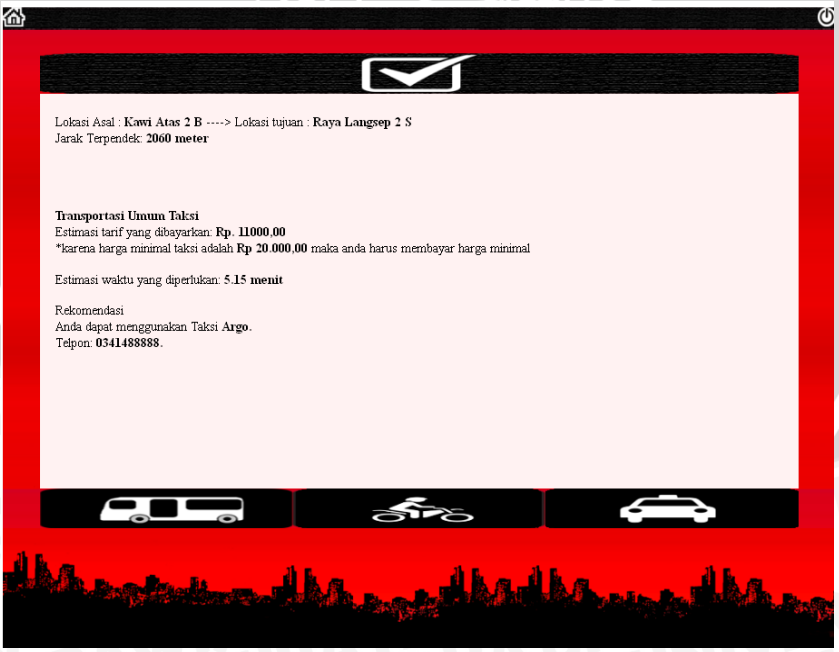
Gambar 5.8 Tampilan Halaman Detail Transportasi Umum Ojek
Sumber: Implementasi

5.5.5 Implementasi Halaman Detail Transportasi Umum Taksi

Pada halaman detail transportasi umum taksi ditampilkan hasil perhitungan estimasi waktu dan tarif yang akan dibayarkan berdasar jarak yang telah diproses dengan algoritma *Floyd-Warshall*. Pada halaman ini juga akan ditampilkan profil taksi yang tersedia di jalan tersebut untuk tujuan *advertisement*. Halaman ini dibuat sebagai alternatif pilihan jika hasil perhitungan sistem pendukung keputusan tidak mengarah pada transportasi umum taksi.

Berikut merupakan cuplikan *script* proses perhitungan estimasi waktu dan tarif transportasi umum taksi:

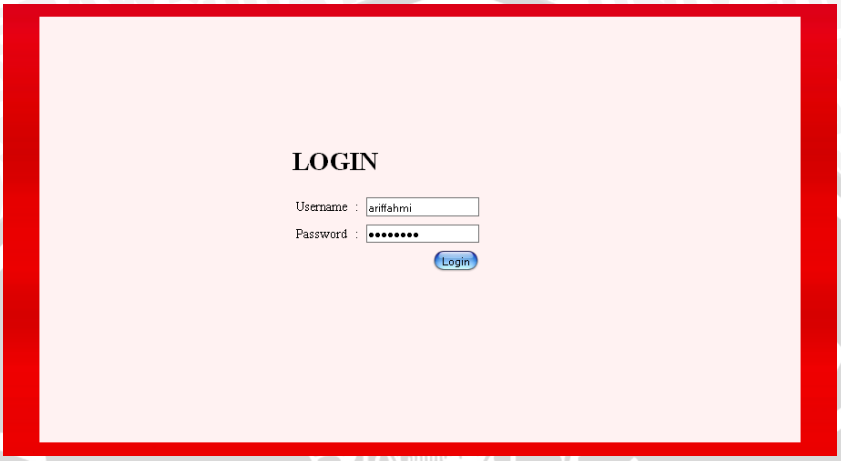
```
//tarif
      echo "Estimasi tarif yang dibayarkan: ";
      $dtpros = $dt-100;
      if ($dtpros>1)
            {
                  $dtpros= $dtpros/100;
                  $dtprosceil= ceil ($dtpros);}
      else
            {$dtprosceil= 0;}
      $ttaksifix= ($dtprosceil*300) + 5000;
      echo "<strong>Rp. $ttaksifix,00 </strong> <br/>";
      if ($ttaksifix <20000) {
            echo "          *karena harga minimal taksi adalah
<strong>Rp  20.000,00</strong>  maka  anda  harus  membayar  harga
minimal<br/><br/>";}

      //waktu
      echo "Estimasi waktu yang diperlukan: ";
      $wtaksi = ($dt/400);
      $wtaksifix = sprintf ("%01.2f", $wtaksi);
      echo "<strong>$wtaksifix menit</strong> <br/>";

      //rekomendasi
      $qreko=mysql_query   ("SELECT   *   FROM   taksiview   WHERE
nama_vertex='$asal'");
      while ($sqlreko = mysql_fetch_row($qreko)){
      echo  "<br/>Rekomendasi  <br/>  Anda  dapat  menggunakan  Taksi
<strong> $sqlreko[2Telpon: <strong>$sqlreko[3].</strong>";}
```



Gambar 5.9 Tampilan Halaman Detail Transportasi Umum Taksi
Sumber: Implementasi

5.5.6 Implementasi Halaman Login *Administrator*

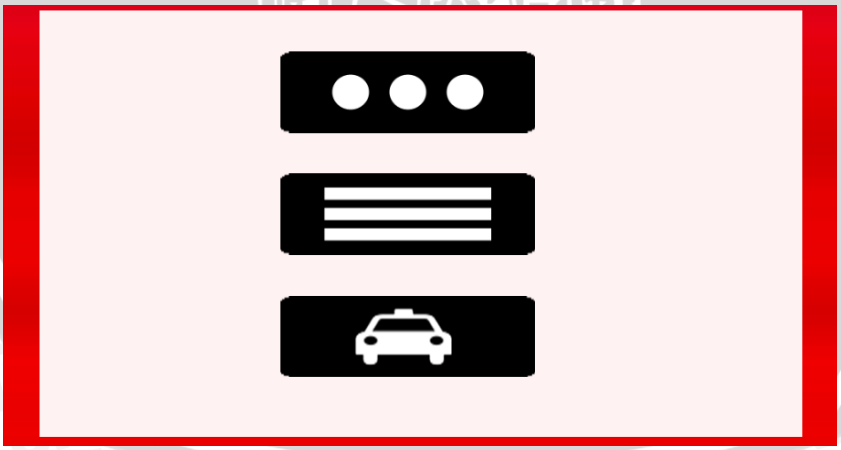
Sebelum melakukan kegiatan *insert*, *edit*, *delete*, *administrator* harus melakukan proses login terlebih dahulu. Pada halaman login, *administrator* memasukkan *username* dan *password* kemudian memilih tombol login.



Gambar 5.10 Tampilan Halaman Login *Administrator*
Sumber: Implementasi

5.5.7 Implementasi Halaman *Home Administrator*

Halaman ini tampil setelah *admin* melakukan proses login. Ada tiga kategori data yang dapat diolah oleh *admin*, yaitu node, vertex, dan transportasi. Tampilan halaman *home administrator* diperlihatkan pada **Gambar 5.11**.



Gambar 5.11 Tampilan Halaman *Home Administrator*
Sumber: Implementasi

5.5.8 Implementasi Halaman Kelola *Node*

Di halaman ini, *admin* dapat memilih beberapa fitur yang berhubungan dengan kelola data node. Fitur tersebut antara lain *view node* dan *add node*.

Gambar 5.12 Tampilan Halaman Kelola *Node*
Sumber: Implementasi

5.5.9 Implementasi Halaman Kelola *Vertex*

Di halaman ini, *admin* dapat memilih beberapa fitur yang berhubungan dengan kelola data *vertex*. Fitur tersebut antara lain *view vertex, add vertex, edit vertex,* dan *delete vertex*

Gambar 5.13 Tampilan Halaman Kelola *Vertex*
Sumber: Implementasi

5.5.10 Implementasi Halaman Kelola Transportasi

Di halaman ini, *admin* dapat memilih beberapa fitur yang berhubungan dengan kelola data transportasi. Fitur tersebut antara lain *add* angkot, *add vertex to* trayek, *add* ojek, *add* ojek *to vertex*, *add* taksi, dan *add* taksi *to vertex*.

Gambar 5.14 Tampilan Halaman Kelola Transportasi
Sumber: Implementasi