SISTEM INFORMASI OPTIMASI RUTE PENGIRIMAN BARANG MENGGUNAKAN ALGORITMA BELLMAN FORD (STUDI KASUS : JNE CABANG JEMBER)

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Pendidikan Sarjana (S1) Fakultas Ilmu Komputer Universitas Jember dan mencapai gelar Sarjana Komputer

Oleh
Rakhmaddian Mubarak
NIM 122410101081

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS JEMBER
2019

SISTEM INFORMASI OPTIMASI RUTE PENGIRIMAN BARANG MENGGUNAKAN ALGORITMA BELLMAN FORD (STUDI KASUS : JNE CABANG JEMBER)

SKRIPSI

diajukan guna melengkapi tugas akhir dan memenuhi salah satu syarat untuk menyelesaikan Pendidikan Sarjana (S1) Fakultas Ilmu Komputer Universitas Jember dan mencapai gelar Sarjana Komputer

Oleh
Rakhmaddian Mubarak
NIM 122410101081

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS ILMU KOMPUTER
UNIVERSITAS JEMBER
2019

PERSEMBAHAN

Skripsi ini saya persembahkan untuk

1. Allah SWT yang telah memberikan rahmad dan hidayahNya memberikan akal dan kelancaran dalam mengerjakan skripsi ini
2. Seluruh keluarga besar Fakultas Ilmu Komputer
3. Orang Tua yang selalu mendukung saya dalam mengerjakan skripsi
4. Adik adik kandung saya :
   Rakhmadyah Raras Arum
   Rakhmagfiroh Geonina Ganestri
   Rakhmawati Oktasenta Safira
   Rakhmaluna Dewi Pramesti
5. Sahabatku yang menemani dari semester awal sampai semester akhir

MOTTO

Jangan pernah berpaling dari kehidupan! Karena era akan terus berganti!

(Edward Newgate)

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Rakhmaddian Mubarak

NIM : 122410101081

menyatakan dengan sesungguhnya bahwa karya ilmiah yang berjudul "Sistem Informasi Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford", adalah hasil karya sendiri, kecuali jika dalam pengutipan substansi disebutkan sumbernya, belum pernah diajukan pada institusi mana pun, dan bukan karya jiplakan. Saya bertanggung jawab atas keabsahan dan kebenaran isinya sesuai dengan sikap ilmiah yang harus dijunjung tinggi.

Demikian pernyataan ini saya buat dengan sebenarnya, tanpa ada tekanan dan paksaan dari pihak manapun serta bersedia mendapat sanksi akademik jika di kemudian hari pernyataan ini tidak benar.

Jember,

Yang menyatakan,

Rakhmaddian Mubarak

NIM 122410101081

SISTEM INFORMASI OPTIMASI RUTE PENGIRIMAN BARANG MENGGUNAKAN ALGORITMA BELLMAN FORD (STUDI KASUS : JNE CABANG JEMBER)

Oleh
Rakhmaddian Mubarak
NIM 12410101081

Pembimbing :

Dosen Pembimbing Utama	: Prof. Drs. Slamin, M.Comp.Sc., Ph.D.
Dosen Pembimbing Pendamping	: Fahrobby Adnan S.Kom., M.MSI

PENGESAHAN PEMBIMBING

Skripsi berjudul "Sistem Informasi Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford" telah diuji dan disahkan pada:

Hari, tanggal : Kamis, 12 Juli 2019

Tempat : Fakultas Ilmu Komputer Universitas Jember

Disetujui oleh :

Pembimbing I,	Pembimbing II,
Prof. Drs. Slamin, M.Comp.Sc., Ph.D.	Fahrobby Adnan S.Kom., M,MSI
NIP 196704201992011001	NIP 198706192014041001

PENGESAHAN PENGUJI

Skripsi berjudul "Sistem Informasi Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford" telah diuji dan disahkan pada:

Hari, Tanggal : Kamis, 12 Juli 2019

Tempat : Fakultas Ilmu Komputer, Universitas Jember.

Tim penguji:

Penguji I,	Penguji II,
Drs.Antonius Cahya P,M.App., Sc.,Ph.D	Priza Pandunata, S.Kom., M.Sc
NIP. 196909281993021001	NIP. 19830131201504001

Mengesahkan

a.n Dekan

Wakil Dekan I Fakultas Ilmu Komputer

Drs. Antonius Cahya P, M.App.Sc., Ph.D.

NIP. 196909281993021001

PRAKATA

Puji syukur kehadirat Allah SWT atas segala rahmat dan karunia-Nya, sehingga penulis dapat menyelesaikan skripsi dengan judul Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford. Skripsi ini disusun untuk memenuhi salah satu syarat menyelesaikan pendidikan Strata Satu (S1) pada Program Studi Sistem Informasi Universitas Jember.

Penyusunan skripsi ini tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis menyampaikan terima kasih kepada:

1. Prof. Dr. Saiful Bukhori, ST., M.Kom., selaku Ketua Program Studi Sistem Informasi Universitas Jember;
2. Prof. Drs. Slamin, M.Comp.Sc., Ph.D., selaku Dosen Pembimbing Utama dan Fahrobby Adnan, S.Kom., M.MSI., selaku Dosen Pembimbing Anggota yang telah meluangkan waktu, pikiran, dan perhatian dalam penulisan skripsi;
3. Seluruh Bapak dan Ibu dosen beserta staf karyawan di program studi sistem informasi;
4. Seluruh keluarga besar RAKHMA tersayang
5. Supervisor JNE Cabang Jember yang meluangkan waktunya dalam membantu penyelesaian skripsi ini;
6. Sahabat dan keluarga besar FORMATION angkatan 2012 yang telah menjadi keluarga selama menempuh pendidikan S1;

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, oleh sebab itu penulis mengharapkan adanya masukan yang bersifat membangun dari semua pihak. Penulis berharap skripsi ini dapat bermanfaat bagi semua pihak.

Jember, 06 Juni 2018

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

Halaman

BAB 1. PENDAHULUAN

1.1. Latar Belakang

JNE merupakan perusahaan yang bergerak dalam bidang pengiriman yang berpusat di Jakarta, Indonesia. JNE merupakan singkatan dari Jalur Nugraha Ekakurir yang memulai kegiatan usahanya pada penanganan kegiatan impor pengiriman barang dan dokumen. JNE memiliki peran sebagai jasa pengiriman barang dan dokumen dengan tujuan membantu memenuhi kebutuhan masyarakat dalam hal pengiriman barang jarak jauh sesuai dengan alamat tujuan yang telah ditentukan. Pengiriman barang dilakukan saat jam kerja. Waktu pengiriman barang dipengaruhi oleh jarak kota yang dituju, semakin jauh jarak yang ditempuh maka pengiriman barang akan semakin lama. Pengguna JNE dapat mengetahui keberadaan barang kiriman melalui fitur *tracking* pada sistem JNE yang telah disediakan. JNE bekerja sama dengan toko-toko online yang ada di Indonesia. Toko-toko online tersebut mempercayakan pengiriman barang kepada konsumen dengan menggunakan jasa pengiriman JNE.

Barang yang akan dikirim akan melalui beberapa tahap, yaitu tahap pengumpulan, pengecekan dan pengiriman. Tahap pengumpulan yaitu proses penampungan barang pada *Warehouse* kantor JNE. Tahap kedua yaitu tahap pengecekan barang seperti kota tujuan dan jenis barang. Tahap terakhir adalah tahap pengiriman barang, tahap ini dilakukan setelah melalui tahap pengumpulan dan pengecekan, pada tahap pengiriman barang, barang tersebut akan dikirim kepada alamat penerima.

Pada saat kurir melakukan pengiriman barang terdapat beberapa hal yang kurang efisien didalam segi kecepatan pengiriman. Beberapa konsumen mengeluh karena barang tidak cepat sampai di tempat tujuan. Penyebabnya adalah kurir tidak efisien didalam memilih atau menentukan jalan yang akan dilewati saat melakukan pengiriman barang dan kurir belum mengetahui rute mana saja yang jaraknya dekat untuk dilewati.

Pada permasalahan diatas, terdapat sebuah algoritma yang dapat membantu didalam menentukan rute terpendek yang akan dilewati oleh kurir, yaitu Algoritma Bellman Ford. Algoritma Bellman Ford adalah salah satu algoritma didalam teori graf yang bertujuan untuk mencari sisi terpendek didalam sebuah lintasan. Algoritma ini dapat diterapkan untuk mencari rute terpendek saat melakukan pengiriman barang. Tujuan dari algoritma ini adalah untuk memudahkan kurir dalam menentukan sebuah rute yang akan dipilih. Algoritma tersebut akan diterapkan dalam sebuah aplikasi yaitu aplikasi optimasi rute pengiriman barang.

Dengan adanya aplikasi optimasi rute pengiriman barang ini, diharapkan memudahkan kurir dalam melakukan pengiriman barang. Aplikasi berbasis web ini berfungsi untuk mengatahui rute yang akan dilalui oleh kurir disaat melakukan pengiriman barang. Aplikasi yang dibangun berbasis web karena pihak JNE masih dalam tahap penyesuaian penggunaan IT pengiriman barang ini dan biaya yang diperlukan relatif murah. Pada aplikasi yang akan dibangun, menerapkan sebuah algoritma, yaitu Algoritma Bellman Ford. Algoritma Bellman Ford disini berperan sebagai metode untuk menghitung dan mencari rute terpendek dari beberapa jumlah pilihan rute yang ada. Cara perhitungan dari metode Bellman Ford adalah dengan menghitung jumlah titik tujuan atau alamat yang akan dituju, kemudian menghitung jumlah garis atau jarak antar titik tersebut. Jarak antara titik satu dengan titik lain pasti berbeda, pada perhitungan ini Algoritma Bellman Ford akan bekerja. Setelah semua perhitungan selesai, kurir akan mengetahui rute terpendek yang harus dilalui untuk melakukan pengiriman barang.

1.2. Rumusan Masalah

Berdasarkan dari beberapa permasalahan yang telah diuraikan diatas, Maka dapat diambil rumusan masalah sebagai berikut :

1. Bagaimana cara mengimplementasikan Algoritma Bellman Ford dalam mencari jalur terpendek yang akan dilalui?

2. Bagaimana cara merancang aplikasi untuk optimasi rute pengiriman barang menggunakan Algortima Bellman Ford?

1.3. Tujuan dan Manfaat

1.3.1 Tujuan

Tujuan dari penelitian ini adalah :

1. Untuk merancang dan membangun aplikasi optimasi rute pengiriman barang menggunakan Algoritma Bellman Ford.
2. Untuk mempermudah kurir dalam proses pengiriman barang

1.3.2 Manfaat

Manfaat dari penelitian ini adalah :

1. Manfaat Akademis

   Hasil penelitian ini diharapkan dapat memberikan kontribusi dan masukan bagi siapa saja yang membutuhkan informasi yang berhubungan dengan judul penelitian ini. Selain itu, hasil penelitian ini merupakan suatu upaya untuk menambah varian judul penelitian yang ada di Program Studi Sistem Informasi Universitas Jember.

2. Manfaat bagi peneliti

   Mengetahui bagaimana cara mengimplementasikan Algoritma Bellman Ford untuk memperpendek jarak pada sebuah rute.

1.4. Batasan Masalah

Terdapat beberapa batasan masalah yang diangkat sebagai parameter pengerjaan penelitian ini diataranya sebagai berikut :

1. Aplikasi yang dibangun berbasis *web.*
2. *Output* yang dihasilkan adalah sebuah website yang dapat menampilkan jarak terpendek pengiriman barang.
3. Aplikasi yang dibangun tidak dapat menampilkan lokasi pengiriman barang dengan akurat. Radius simpangan kurang lebih satu kilometer.

4. Aplikasi ini hanya mempertimbangkan jarak, tidak termasuk faktor lainnya seperti waktu, kemacetan, dan lain lain.

1.5. Sistematika Penulisan

Adapun sistematika penulisan skripsi ini adalah sebagai berikut:

1. Pendahuluan

Bab ini memuat uraian tentang latar belakang, rumusan masalah, tujuan dan manfaat, batasan masalah, dan sistematika penulisan skripsi yang masing-masing terdapat dalam sub bab tersendiri.

2. Tinjauan Pustaka

Bab ini memaparkan tinjauan terhadap hasil-hasil penelitian terdahulu berkaitan dengan landasan materi, dan kajian teori metode analisis data yang berkaitan dengan masalah dalam penelitian.

3. Metode Penelitian

Bab ini menguraikan tentang jenis penelitian, tahapan desain sistem, model perancangan sistem dan gambaran sistem yang akan dibuat.

4. Pengembangan Sistem

Bab ini berisi uraian tentang deskripsi umum sistem, pengumpulan data, analisis kebutuhan dan desain sistem. Langkah-langkah yang ditempuh dalam proses ini adalah untuk mengetahui rancangan sistem.

5. Hasil dan Pembahasan

Bab ini berisi tentang hasil implementasi algoritma Bellman Ford dalam penentuan rute terpendek pengiriman barang.

6. Penutup

Bab ini terdiri atas kesimpulan atas penelitian yang telah dilakukan dan saran untuk penelitian selanjutnya.

BAB 2. TINJAUAN PUSTAKA

Bab ini memaparkan tinjauan terhadap hasil-hasil penelitian terdahulu berkaitan dengan landasan materi, dan kajian teori metode analisis data yang berkaitan dengan masalah dalam penelitian

2.1. Penelitian Terdahulu

Penelitian terdahulu yang berjudul "Aplikasi Algoritma Bellman Ford dalam Meminimumkan Biaya Operasional Rute Penerbangan", dilakukan oleh Bowo Kristanto, mahasiswa Universitas Islam Negeri Sunan Kalijaga, Fakultas Sains dan Teknologi. Hasil penelitian dan pembahasan pada literatur ini dapat disimpulkan berhasil dalam penerapannya. Permasalahan suatu graf dalam menentukan rute penerbangan pada penelitian ini karena adanya bobot (biaya) yang bernilai negatif. Algoritma Bellman Ford dapat menyelesaikan permasalahan graf tersebut dikarenakan [ada algoritma ini dapat memberikan solusi jalur terpendek yang dapat memberikan dampak positif dalam estimasi biaya operasional penerbangan. Algoritma Bellman Ford dapat diaplikasikan dalam dunia nyata, seperti menentukan rute penerbangan agar dapat meminimumkan biaya operasional bahan bakar.

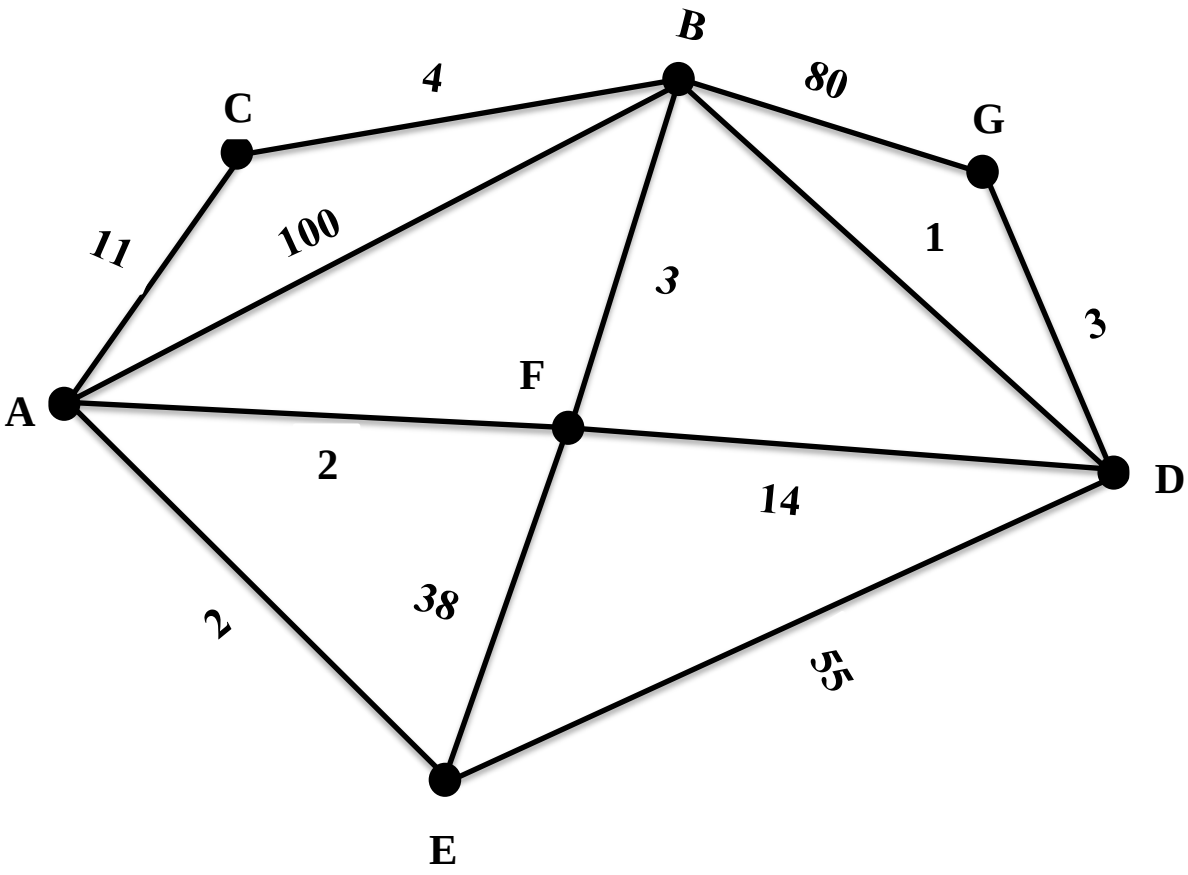
Penelitian terdahulu yang berjudul "Penentuan Jalur Terpendek Menuju Cafe di Kota Malang Menggunakan Metode Bellman Ford dengan Location Based Service Menggunakan Android", yang dilakukan oleh M.Rofiq, Riza Fathul Uzzy, dari STMIK ASIA Malang. Penjelasan yang telah dibahas oleh penulis pada laporan tugas akhir ini, maka dapat diperoleh beberapa kesimpulan. Salah satu tujuan dibangunya sebuah aplikasi Point Cafe ini dapat membantu pengguna dalam memilih cafe yang terdapat di kota Malang dengan lebih mudah. Koordinat cafe akan memiliki tingkat akurasi yang baik apabila pengguna menggunakan aplikasi tersebut di luar ruangan atau tempat terbuka.

2.2. Teori Graf

Graf adalah model untuk merepresentasikan suatu objek diskrit yang membentuk hubungan antar objek-objek tersebut. Graf dapat diartikan sebagai pasangan himpunan (V,E) yang dalam hal ini, V adalah himpunan tidak kosong dari simpul-simpul (vertices atau node) sedangkan E adalah himpunan sisi (*edges atau arcs*) yang menghubungkan sepasang simpul [1].

V = { v1,v2, ... , vn}…….persamaan i
E = { e1,e2, … , en}…….persamaan ii
G = (V,E) (3)……...persamaan iii

Graf memiliki banyak fungsi dalam kehidupan sehari-hari yang dapat diterapkan untuk menyelesai beberapa permasalahan. Salah satu penerapannya adalah dalam pengaturan jaringan komputer. Pada jaringan komputer, node merepresentasikan komputer sedangkan edges merepresentasikan koneksi antar komputer. Graf yang digunakan adalah graf berbobot atau weighted graph. Permasalahan yang dapat diambil adalah cara sebuah jaringan dapat dioptimalkan kerjanya, seperti melakukan transfer data antar komputer satu dengan komputer lain. Ketika jarak antar komputer berbeda secara signifikan maka kemampuan melakukan transfer data antar komputer akan berbeda. Adanya perbedaan pada permasalahan ini maka diperlukan solusi yaitu mencari jalur tercepat atau terpendek untuk mengoptimalkan kinerja jaringan yang telah disusun. Jaringan tersebut biasanya menggunakan graf berbobot untuk menyelesaikannya. Graf berbobot tersebut dapat dilihat pada gambar 2.1 graf berbobot berikut:



Gambar 2.1 Graf Berbobot

(Retanto Yudi, 2009)

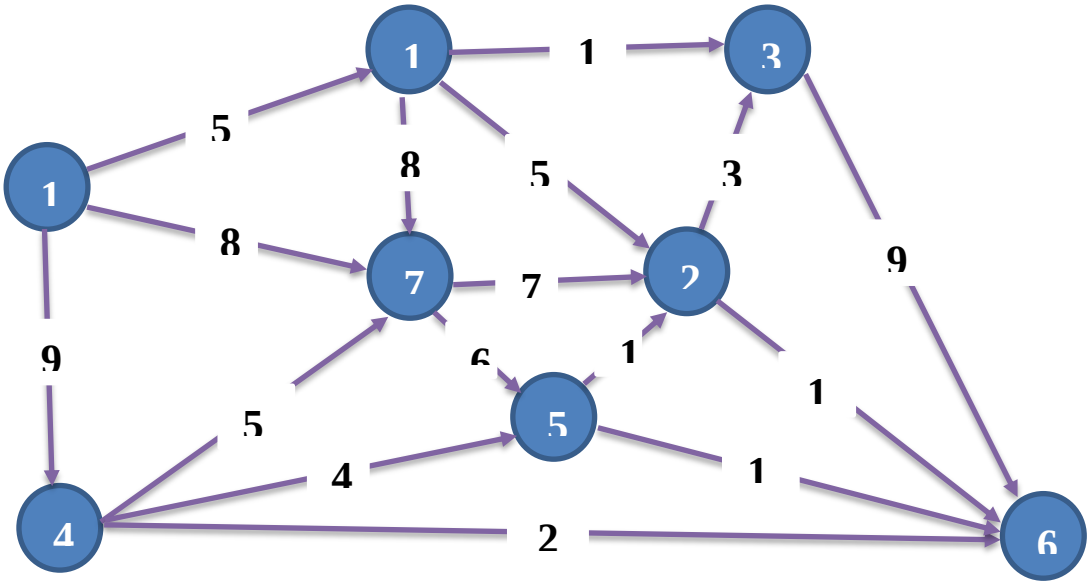
Gambar 2.1 graf berbobot menggambarkan jalur terpendek dari simpul A ke simpul B yang harus melewati simpul F terlebih dahulu. Secara visual pencarian jalur terpendek masih mungkin dilakukan, walaupun akan sangat sulit jika simpul bertambah banyak dan perbedaan jarak antar simpul sangat kecil. Banyaknya perhitungan yang harus dilakukan jika simpul bertambah banyak membuat perhitungan harus dilakukan menggunakan komputer, sehingga dibutuhkan algoritma pencarian jalur terpendek yang akurat. Sebagai mahasiswa program studi sistem informasi, masalah ini dapat diselesaikan dengan merancang sebuah algoritma yang mampu menentukan jalur terpendek antar simpul. Kesalahan perhitungan dapat dihindari sehingga jumlah simpul yang banyak tidak akan menjadi hambatan dalam penentuan

jalur terpendek. Pada kenyataannya sudah ada dua buah algoritma untuk menghitung jarak terpendek, yaitu algoritma Dijkstra dan algoritma Bellman Ford.

2.3. Algoritma Bellman Ford

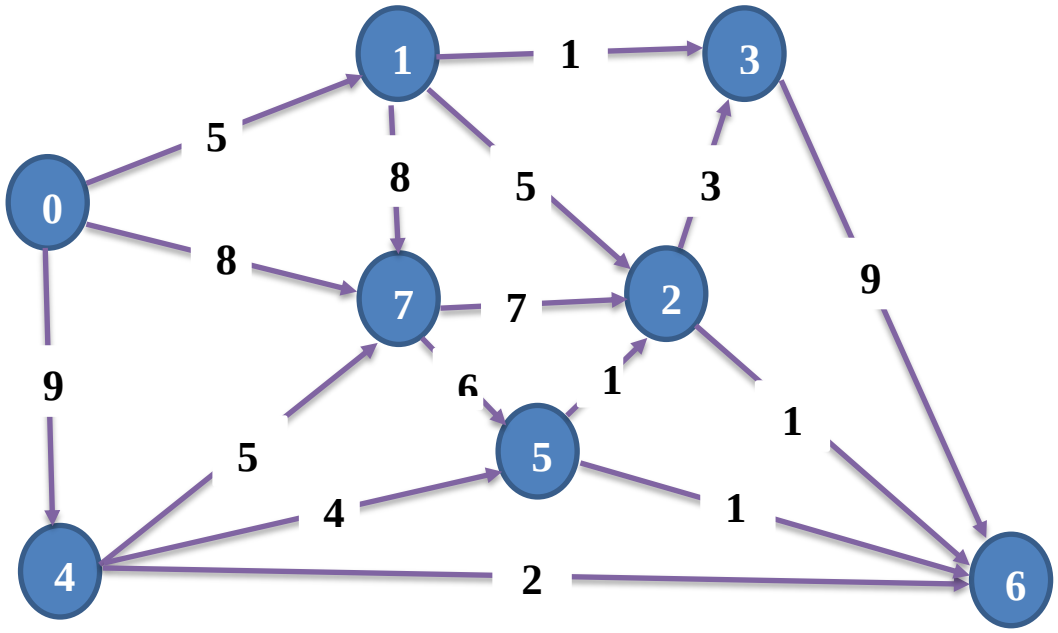
Algoritma Bellman Ford adalah salah satu dari algoritma yang digunakan untuk menghitung jalur terpendek dari graf berbobot.Algoritma ini memiliki 1 kemungkinan, yaitu notasi yang diberikan O(V·E). V adalah jumlah vertex atau simpul dalam graf berbobot,E merupakan edges jumlah sisi dalam graf.Oleh sebab itu untuk jumlah sisi ataupun simpul yang sangat besar akan menyebabkan algoritma ini akan berjalan lebih lama dibandingkan dengan algoritma yang lain.

Contoh pencarian rute terpendek dengan menggunakan algoritma Bellman Ford yang dapat dilihat pada gambar 2.2 algorritma fourth edition sedangkan gambar 2.3 dan 2.4 menjelaskan tentang perhitungan algoritma bellman ford berikut:



Gambar 2.2 Algorithms Fourth Edition

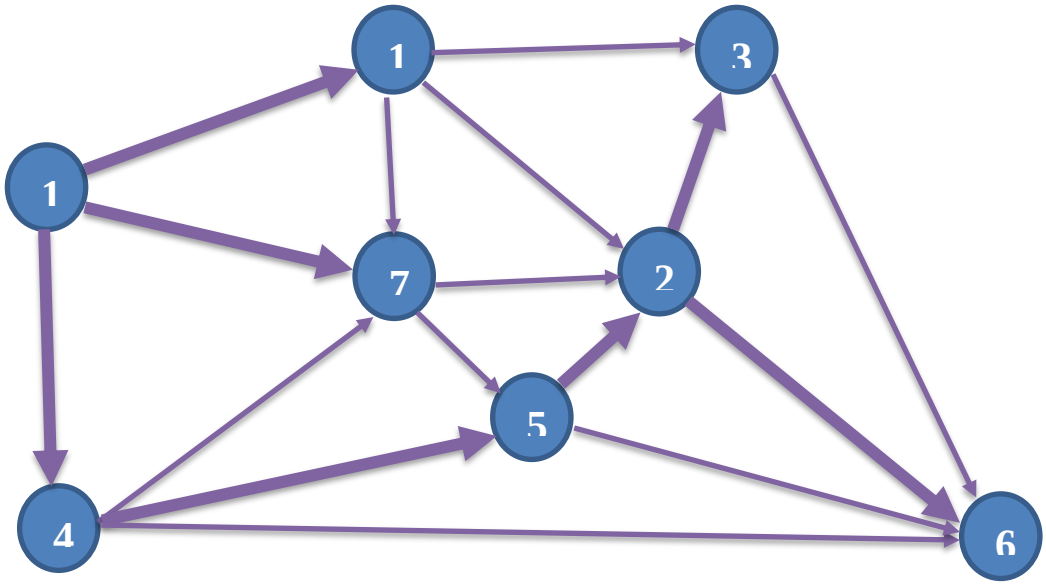
(Sedgewick Robert, Wayne Kevin)



0-1 = 5, **1-3 = 15,** **3-6 = 9,** **5-2 = 1,**
0-7 = 8, **1-7 = 4,** **4-5 = 4,** **5-6 = 13,**
0-4 = 9, **2-3 = 3,** **4-6 = 20,** **7-5 = 6,**
1-2 = 12, **2-6 = 16,** **4-7 = 5,** **7-2 = 7,**

Gambar 2.3 Algorithms Fourth Edition

(Sedgewick Robert, Wayne Kevin)



V	distTo[]	edgeTo[]
0	0.0	-
1	5.0	0-1
2	14.0	5-2
3	17.0	2-3
4	9.0	0-4
5	13.0	4-5
6	25.0	2-6
7	8.0	0-7

Gambar 2.4 Algorithms Fourth Edition

(Sedgewick Robert, Wayne Kevin)

2.4. Rute Terpendek

Rute terpendek adalah rute minimum yang diperlukan untuk suatu tempat dari tempat tertentu. Rute minimum yang dimaksd dapat dicari dengan menggunakan graf.

Graf yang digunakan adalah graf berbobot, Yaitu graf yang setiap sisinya diberikan suatu nilai atau bobot.

Dalam kasus ini yang dimaksud berupa jarak. Dalam hal ini bobot harus bernilai positif, pada lain hal terdapat bobot dengan nilai negatif. Rute terpendek dengan titik awal s dan titik tujuan t didefenisikan sebagai rute dari s ke t dengan bobot minimum dan berupa rute sederhana(simple path). Salah satu aplikasi graf berarah berlabel yang sering dipakai adalah mencari rute terpendek diantara 2 titik. Apabila masalahnya adalah mencari rute terpendek tetap dapat digunakan dengan cara mengganti nilai sisi. Terdapat beberapa jenis persoalan rute terpendek, antara lain:

1.Rute terpendek antara dua simpul tertentu.

2. Rute terpendek antara semua pasangan simpul.

3. Rute terpendek dari simpul tertentu ke semua simpul yang lain.

4. Rute terpendek antara dua buah simpul yang melalui beberapa simpul tertentu Misalkan G adalah suatu graf, untuk v dan w adalah titik dalam G suatu Walk dari v ke w adalah barisan titik dan sisi secara berselang-seling, diawali dari titik v dan diakhiri pada titik w. Walk dengan panjang n dari v ke w ditulis : v0 e1 v1 e2 v2 … vn-2 en vn dengn v0 = v: vn = w; vi-1 dan vi adalah titik-titik ujung sisi ei .Lintasan dengan panjang n dari v ke w adalah walk dari v ke w yang semua sisinya berbeda. Rute dari v ke w dituliskan sebagai v = v0 e1 v1 e2 v2 … vn-1en vn = w dengan ei ≠ ejuntuk i ≠ j. Penulisan berikutnya akan dipergunakan notasi v1 v1, A = {v1 v2, v2 v3, ... }.

2.5. Langkah-Langkah Algoritma

Langkah-langkah untuk menyelesaikan lintasan terpendek dengan menggunakan Algoritma Bellman-ford adalah:

a. Menentukan titik asal dan mendaftar semua titik maupun sisi

b. Memberi nilai untuk titik asal sama dengan nol dan titik-titik lainnya dengan tak hingga.

c. Memulai iterasi terhadap semua titik yang ada dimulai dengan titik asal. untuk menentukan jarak dari semua titik yang berhubungan dengan titik asal dengan formula seperti berikut: U = titik asal V = titik tujuan UV = sisi yang menghubungkan U dan V Jika jarak V lebih besar dari jarak U+bobot UV maka jarak V diisi dengan jarak U+bobot UV, dilakukan hingga semua titik terjelajahi. Melakukan iterasi untuk semua sisi yang ada untuk mengecek apakah ada siklus negatif dalam graf tersebut, kemudian melakukan pengecekan seperti dibawah ini: Untuk semua sisi UV, jika jarak V lebih besar dari jarak U+bobob UV maka sudah jelas bahwa graf tersebut memiliki siklus *negative.*

Keterangan:

U = vertex asal

V = vertex tujuan

UV = Edges yang menghubungkan U dan V

Jika distance V, lebih kecil dari distance U + weight UV maka distance V, diisi dengan distance U + weight UV

Lakukan hingga semua vertices terjelajahi

BAB 3. METODOLOGI PENELITIAN

Bab ini menguraikan tentang jenis penelitian, waktu penelitian, tahap penelitian, tahap pengumpulan data, tahap pengolahan dan implementasi sistem yang akan dibuat pada penelitian ini.

3.1. Jenis Penelitian

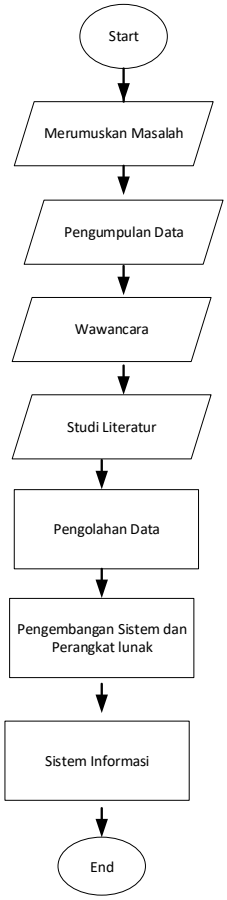
Jenis penelitian yang digunakan pada penelitian ini adalah pengembangan, tujuan dari penelitian ini adalah untuk membangun sebuah sistem informasi. Penelitian ini digunakan untuk menghitung dengan menggunakan metode Bellman Ford yang diterapkan dalam proses perhitungan rute terpendek.

3.2. Waktu Penelitian

Tempat pelaksanaan penelitian ini bertempat di JNE Kantor Cabang Utama Jember, jalan Moh.Yamin No. 99 dengan rentang waktu bulan April 2018 hingga bulan Mei 2018.

3.3. Tahapan Penelitian

Tahapan tahapan yang dilakukan dalam penelitian ini dapat dilihat pada gambar berikut:



Gambar 3.1 *Flowchart* Tahapan Penelitian

3.3.1 Merumuskan Masalah

Merumuskan masalah menjadi dasar pemikiran dalam penelitian ini. Untuk lebih jelas mengenai rumusan masalah dari penelitian ini dapat dilihat pada subbab 1.2.

3.3.2 Pengumpulan Data

Pengumpulan data adalah proses yang dibutuhkan untuk melakukan penelitian dan acuan untuk membuat sistem dan aplikasi.

3.3.3 Wawancara

Wawancara adalah pengumpulan data berupa tanya jawab peneliti dengan narasumber. Wawancara berupa percakapan langsung antara satu atau beberapa pihak yang bertujuan mendapatkan informasi secara lisan.

3.3.4 Studi Literatur

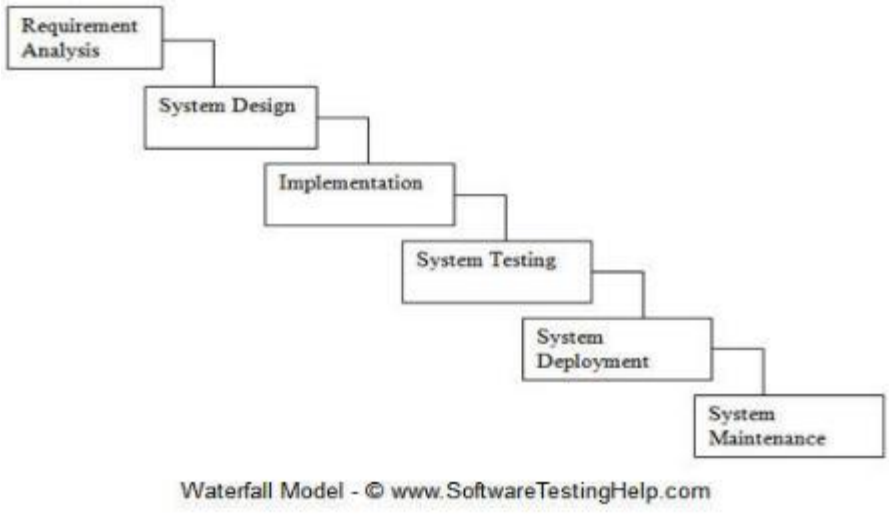
Studi literatur berisi tentang teori dan bahan penelitian yang diperoleh dari bahan acuan untuk dijadikan landasan kegiatan penelitian.Studi literatur berisi tentang ulasan , rangkuman dan penulisan beberapa sumber pustaka (berupa artikel, buku, slide informasi dari internet) berisi tentang topik yang dibahas.

3.3.5 Tahap Pengolahan Data

Data yang telah diperoleh pada tahap pengumpulan data diolah untuk menentukan kebutuhan-kebutuhan atau fungsionalitas sistem informasi pengendalian persediaan bahan baku menggunakan algoritma Bellman Ford.

3.3.6 Pengembangan Sistem dan Perangkat Lunak

Pada tahap ini menjelaskan tentang alur pembuatan sistem dan perangkat lunak dan model SDL waterfall. Model yang digunakan dalam penelitian ini adalah model waterfall. Alur kerja waterfall ditunjukkan pada gambar 3.2



Gambar 3.2 Model *Waterfall*

1. *Requirement Analysis*

Requirement analysis merupakan kebutuhan apa saja yang akan digunakan untuk melakukan penelitian, seperti wawancara dan studi literatur

2. *System Design*

Spesifikasi yang dibutuhkan dari tahap pertama dipelajari pada tahap ini dan disiapkan untuk desain sistem. Desain sistem membantu dalam menganalisa spesifikasi hardware dan kebutuhan sistem dan juga membantu dalam mendefinisikan arsitektur sistem secara keseluruhan.

Tahapan berikutnya adalah desain sistem menggunakan Unifed Modeling Languange (UML) yang dirancang menggunakan konsep Object-Oriented Programming (OOP). Berikut adalah pemodelan UML yang digunakan amtara lain:

a. *Business Process*

Business Process merupakan diagram yang menggambarkan proses yang lengkap. Isi dari *business process* adalah *resources* dan informasi yang dibutuhkan pada penelitian ini, *event* yang mendorong terjadinya proses dan *goal* yang dituju.

b. *Use Case Diagram*

Use case adalah model yang menggambarkan fungsi atau tugas yang dilakukan oleh user, baik manusia maupun mesin / komputer. *Use Case model* ini dapat digunakan untuk menggambarkan *job spesification* dan *job description*, serta keterkaitan antar *job*.

c. *Scenario*

Scenario Diagram berfungsi untuk menjelaskan alur sistem dari fitur yang ada di *job specification* dan *job description* yang ada pada *Use Case Diagram*. *Scenario* menjelaskan alur sistem dan keadaan yang akan terjadi ketika terjadi suatu *event* tertentu.

d. *Activity Diagram*

Activity Diagram digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi. *Activity diagram* mempunyai fungsi yang sama dengan *scenario* namun diimplementasikan dalam diagram alir.

e. *Sequence Diagram*

Sequence diagram menggambarkan aliran logika interaksi antar objek yang mengindikasikan komunikasi antar obyek di dalam sistem yang disusun pada suatu urutan atau rangkaian waktu.

f. *Class Diagram*

Class diagram menggambarkan struktur dan deskripsi class serta hubungannya antar class, sehingga memudahkan dalam proses pengkodean.

g. *Entity Relationship Diagram (ERD)*

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan data yang mempunyai hubungan antar relasi.

3. *Implementation*

Dengan input dari System Design, sistem pertama kali dikembangkan dalam program yang disebut dengan Unit, yang diintegrasikan dengan tahap berikutnya. Setiap unit dikembangkan dan diuji fungsionalitasnya dan dijadikan referensi sebagai unit testing.

Tahap implementasi dilakukan berdasarkan desain sistem yang selanjutnya diubah dalam bentuk program yaitu:

a. Penulisan program menggunakan Bahasa pemrograman *Page Hyper Text Processor* (PHP) dengan framework Codeigniter *(CI)*
b. *Database Management System (DBMS)* yang digunakan adalah *MySQL* dengan menggunakan jaringan local aplikasi *XAMPP*.

4. *System Testing*

Semua unit yang dikembangkan dalam tahap implementation diintegrasikan kedalam sebuah sistem setelah menguji setiap unit. Pasca integrasi, seluruh sistem diuji untuk setiap kesalahan dan kegagalan.

Tahap pengujian sistem bertujuan untuk mengetahui sejauh mana sistem ini dapat berjalan. Integration Testing berfungsi untuk mengetahui apakah sistem ini dapat berfungsi dengan baik sesuai dengan yang diharapkan, serta untuk mengetahui letak kekurangan pada sistem. Penelitian ini melakukan pengujian sistem dengan cara berikut:

a. *White Box Testing*

White box testing merupakan cara pengujian dengan melihat modul yang telah dibuat dan program-program yang ada. Pengujian ini dilakukan oleh pembuat program *(developer)*. Jika ada modul yang menghasilkan output yang tidak sesuai, maka baris-baris program, variabel dan parameter yang terlibat pada unit tersebut satu persatu akan dicek dan diperbaiki.

b. *Black Box Testing*

Black box testing merupakan metode pengujian perangkat lunak yang memeriksa fungsional dari aplikasi yang berkaitan dengan struktur internal atau kerja. Metode ini memfokuskan pada keperluan fungsional dari software.

5. *Deployment of System*

Setelah pengujian fungsional dan non-fungsional dilakukan, produk dideploy di lingkungan client atau dirilis ke pasar.

6. *Maintenance*

Terdapat beberapa masalah yang muncul di lingkungan client, untuk memperbaikinya maka dirilis patch-nya, juga untuk meningkatkan kualitas produk maka akan dirilis beberapa versi terbaik. Maintenance dilakukan untuk memberikan perubahan perbaikan ini kepada pelanggan.

Tahap pemeliharaan dilakukan ketika sistem memiliki kesalahan yang belum terdeteksi sebelumnya, sehingga kesalahan-kesalahan sistem perlu diperbaiki. Pemeliharaan juga dilakukan apabila ssitem mengalami perubahan-perubahan karena permintaan baru dari user.

BAB 4. PENGEMBANGAN SISTEM

Bab ini akan membahas tentang Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford. Tahap pengembangan sistem dilaksanakan berdasarkan SDLC *waterfall*, dengan uraian mengenai analisa kebutuhan, desain, implementasi, dan pengujian sistem.

4.1 Deskripsi Umum Sistem

Deskripsi umum dari tentang Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford yang dibangun dalam penelitian ini akan dijelaskan lebih detail pada SOP (*statement of purpose*) sistem dan fungsi sistem.

4.1.1 SOP (*Statement of purpose*)

Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford merupakan aplikasi berbasis web yang memiliki beberapa fitur atau fungsi. Fitur tersebut antara lain yaitu mengelola data pengiriman barang, kemudian akan dihitung total jarak dengan menggunakan metode algoritma ballman ford yang merupakan salah satu bagian dari teori graf. Keluaran yang dihasilkan yaitu berupa rute terpendek pengiriman barang. Pada aplikasi ini memiliki dua tingkat hak akses yaitu admun dan kurir. Admin dapat mengelola data barang dan data kurir serta melihat hasil dari perhitungan rute terpendek. Sedangkan kurir hanya dapat melihat hasil dari rute terpendek pengiriman barang tersebut.

4.2 Pengumpulan Data

Pengumpulan data menggunakan data hasil wawancara dengan supervisor JNE Kantor Pusat Jember yang bertempat di Jl. Muhammad Yamin No. 99, Tegal Besar, Kaliwates, Kabupaten Jember, Jawa Timur. *Point* penting yang dihasilkan dari wawancara adalah pengiriman barang yang dilakukan oleh pihak JNE cenderung masih manual sehingga dapat menyebabkan kurang efisiensi terhadap pengeluaran biaya BBM.

4.3 Analisis Kebutuhan

Berdasarkan hasil analisa data yang diperoleh dari wawancara, diperoleh beberapa data barang dan data kurir. Tahapan ini dilakukan analisis terhadap kebutuhan-kebutuhan dari sistem yang dibangun, baik berupa kebutuhan fungsional maupun kebutuhan non-fungsional. Hasil analisis tersebut sangat mempengaruhi fungsionalitas sistem yang dibangun untuk dapat digunakan sesuai dengan fungsi dan kebutuhan pengguna.

4.3.1 Kebutuhan Fungsional

Kebutuhan fungsional sistem berisi fitur-fitur inti yang harus dipenuhi dalam sistem agar sistem mampu difungsikan sesuai dengan tujuan dan kebutuhan pengguna terhadap sistem itu sendiri. Kebutuhan fungsional dari Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford yaitu:

a. Sistem mampu mengelola data barang (*view, insert, delete*)
b. Sistem mampu mengelola data kurir (*view, insert, delete*)
c. Sistem mampu menghasilkan rute terpendek pengirimn barang

4.3.2 Kebutuhan Non-Fungsional

Kebutuhan non-fungsional merupakan fitur-fitur yang dimiliki untuk mendukung sistem dalam memenuhi fungsionalitasnya untuk dapat memenuhi kebutuhan dari pengguna. Kebutuhan non-fungsional dari Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford yaitu:

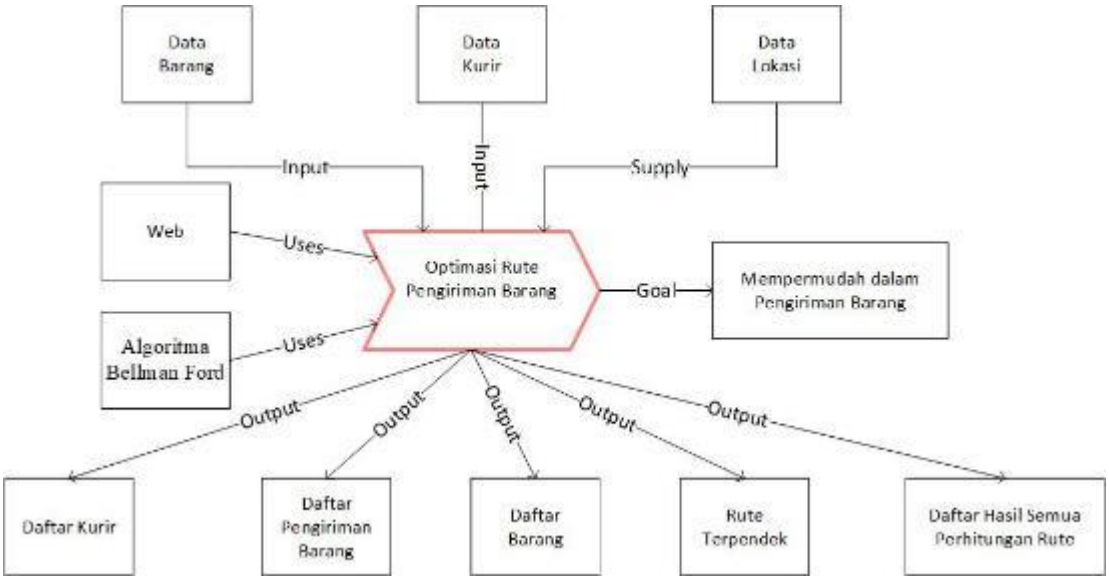
a. Sistem bekerja sesuai dengan fungsinya dan dapat dijalankan pada semua komputer dan *browser* yang berbeda.
b. Tampilan dan bahasa komunikasi sistem mudah dimengerti oleh *user* untuk memberikan kenyamanan pemakaian dan memudahkan pengoperasian
c. Sistem menggunakan *email address* dan *password* untuk autentifikasi akses pengguna terhadap sistem.

4.4 Desain Sistem

Tahapan yang dilakukan setelah melakukan analisis kebutuhan sistem yaitu tahap perencanaan pembangunan sistem yang dapat digambarkan dengan desain sistem. Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford meliputi *business process*, *use case diagram*, *scenario*, *activity diagram*, *sequence diagram*, *class diagram*, dan *entity relationship diagram*.

4.4.1 *Business Process*

Selain dapat dideskripsikan dalam sebuah SOP (*Statement of Purpose*), gambaran umum Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford dapat digambarkan melalui sebuah *business process*. Seperti yang dapat kita lihat pada Gambar 4.1 menggambarkan data-data yang digunakan sebagai masukan, data keluaran, uses sistem yang dibangun, hingga goal dari dibangunnya sistem itu sendiri.

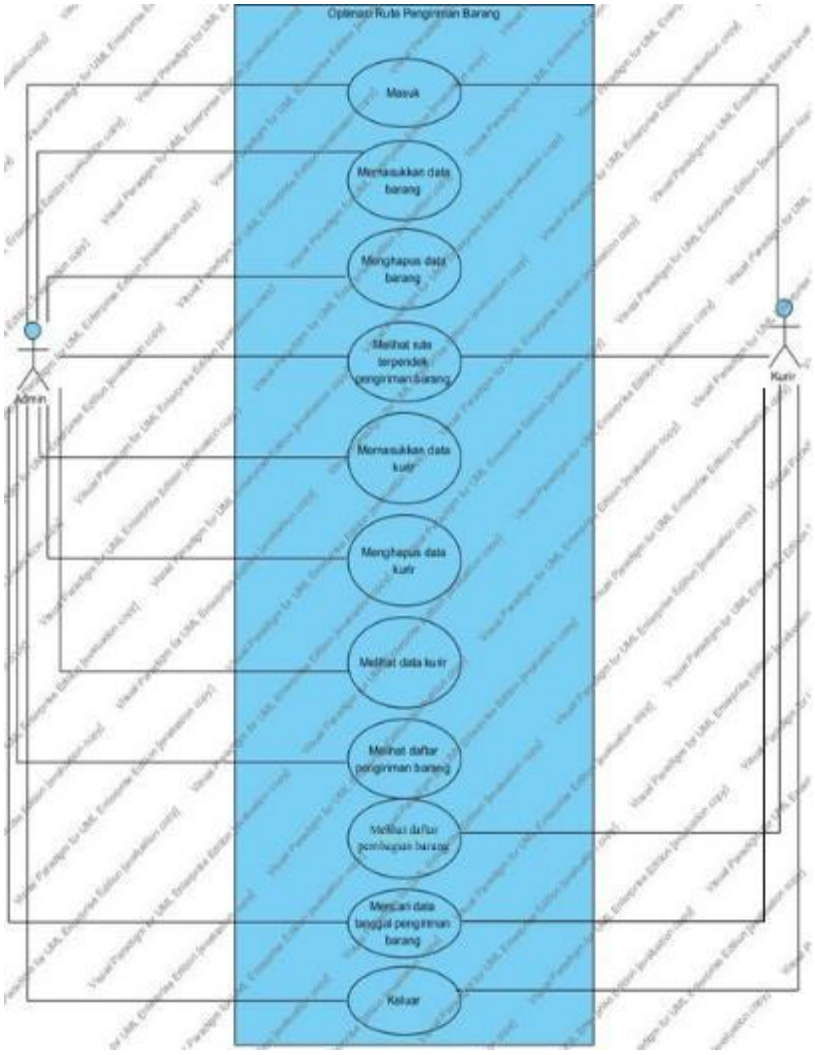


Gambar 4. 1 *Business Process System*

4.4.2 *Use Case* Diagram

Use case diagram merupakan pemodelan yang dibuat untuk dapat menggambarkan interaksi antara aktor Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford yang akan dibangun. Melalui *use case* diagram dapat diketahui interaksi yang dapat dilakukan aktor terhadap sistem sesuai dengan hak

akses yang dimiliki oleh masing-masing aktor atau pengguna. Pada Gambar 4.2 digambarkan *use case* diagram yang terdiri atas dua aktor dengan sembilan *use case*.



Gambar 4. 2 *Use Case* Diagram

Berdasarkan *use case* diagram pada Gambar 4.2 terdapat dua aktor atau pengguna, yaitu admin dan sekertaris. Adapun deskripsi dari masing-masing aktor dapat dilihat pada Tabel 4.1.

Tabel 4.1 Deskripsi pembagian aktor sistem

No.	Aktor	Deskripsi
1	Admin	Admin merupakan aktor yang memiliki hak akses untuk mengelola data barang dan data kurir. Admin dapat memasukkan dan menghapus data barang. Selain itu admin juga dapat memasukkan dan menghapus data kurir. Terakhir admin dapat melihat hasil perhitungan rute terpendek pengiriman barang.
2	Kurir	Aktor ini dapat melihat hasil perhitungan rute terpendek pengiriman barang.

Selain memiliki dua aktor, dalam *use case* diagram juga terdapat delapan *use case*. Deskripsi dari *use case* tersebut dapat dilihat pada Tabel 4.2.

Tabel 4.2 Deskripsi *use case* sistem

No	*Usecase*	Deskripsi
1	Masuk	*Usecase* yang menggambarkan proses masuk ke dalam hak akses sistem masing masing actor
2	Memasukkan data barang	*Usecase* yang menggambarkan proses menambah data barang oleh admin
3	Menghapus data barang	*Usecase* yang menggambarkan proses menghapus data barang oleh admin
4	Melihat rute terpendek	*Usecase* yang menggambarkan proses melihat rute pengiriman barang yang dapat dilihat oleh admin dan kurir

	pengiriman barang	
5	Memasukkan data kurir	*Usecase* yang menggambarkan proses menambah data kurir oleh admin
6	Menghapus data kurir	*Usecase* yang menggambarkan proses menghapus data kurir oleh admin
7	Melihat Data kurir	*Usecase* yang menggambarkan proses melihat data kurir oleh admin
8	Melihat daftar pengiriman barang	*Usecase* yang menggambarkan proses bagi admin untuk melihat daftar pengiriman barang keseluruhan yang akan dikirimkan
9	Melihat daftar pembagian barang	*Usecase* yang menggambarkan proses bagi kurir untuk melihat daftar pembagian barang yang akan dikirimkan
10	Mencari data tanggal pengiriman barang	*Usecase* yang menggambarkan proses bagi admin dan kurir untuk mencari tanggal pengiriman barang yang dikirimkan
11	Keluar	*Usecase* yang menggambarkan proses bagi aktor untuk keluar dari sistem

4.4.3 *Use Case* Skenario

Use case skenario adalah dokumentasi terhadap kebutuhan fungsional sistem. *Use case* skenario Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford adalah sebagai berikut:

1. Skenario Masuk

Skenario masuk merupakan penjelasan urutan reaksi aktor dan reaksi sistem pada skenario normal dan skenario alternatif *use case* skenario masuk. Pada skenario tersebut dapat dilihat bagaimana alur perjalanan suatu fitur masuk antara aksi yang dilakukan oleh aktor yaitu admin dan kurir ketika akan melakukan masuk, kemudian bagaimana reaksi dari sistem untuk merespon aksi tersebut sehingga aktor telah melakukan masuk pada sistem tersebut. Proses lengkap skenario masuk dapat dilihat pada Tabel 4.3

Tabel 4. 1. *Use case* Skenario Masuk

Nama	Masuk
Aktor	Admin, Kurir
Pre-Kondisi	Aktor belum masuk, halaman masuk
Pra-Kondisi	Aktor sudah masuk, halaman utama
SKENARIO NORMAL "Masuk"	
Aktor	Sistem
1. Membuka halaman website Optimasi Rute Pengiriman Barang	
	2. Menampilkan formulir masuk a) E-Mail Address, varchar (255) b) Password, varchar (255)
3. Mengisi formulir masuk	
4. Menekan tombol hak akses (admin atau karyawan)	

5. Menekan tombol Login	
	6. Menampilkan halaman utama sesuai dengan hak akses
SCENARIO ALTERNATIF “Email dan password tidak diisi	
Aktor	Sistem
5. Menekan tombol Login	
	6.Menampilkan notifikasi “Please fill out this field”
SCENARIO ALTERNATIF “Email dan password salah”	
Aktor	Sistem
5. Menekan tombol Login	
	6.Menampilkan formulir masuk E-Mail Address varchar (255) Password varchar (255)

2. Skenario Memasukkan Data Barang

Skenario memasukkan data barang merupakan salah satu menu yang digunakan oleh admin untuk memasukkan data barang yang akan dikirim ke dalam sistem. Menu ini hanya dapat dilakukan oleh admin.

Tabel 4. 2 *Use Case Scenario* Memasukkan Data Barang

Nama	Memasukkan Data Barang
Aktor	Admin
Pre-Kondisi	Admin berhasil masuk ke halaman utama

Pra-Kondisi	Admin berhasil memasukkan data barang, halaman barang
SKENARIO NORMAL “Memasukkan Data Barang”	
Aktor	Sistem
1. Menekan menu barang	
	2. Menampilkan data barang a. Id b. Nama pengirim c. Nama penerima d. Alamat
3. Menekan tombol tambah barang	
	4. Menampilkan formulir pengisian data barang : a. Nama Pengirim, varchar (50) b. Nama Penerima, varchar (50) c. Alamat, varchar (150) d. Kecamatan, varchar (100) e. Pilih Karyawan, varchar (50)
5. Mengisi formulir data barang	
6. Menekan tombol simpan	
	5.Menampilkan halaman data barang
SCENARIO ALTERNATIF “Tipe data salah”	

Aktor	Sistem
3. Mengisi formulir data barang	
4.Menekan tombol simpan	
	7. Menampilkan formulir pengisian data barang : f. Nama Pengirim, varchar (50) g. Nama Penerima, varchar (50) h. Alamat, varchar (150) i. Kecamatan, varchar (100) j. Pilih Karyawan, varchar (50)

3. Skenario Menghapus Data Barang

Skenario menghapus data barang merupakan menu yang digunakan oleh admin untuk menghapus data barang jika terjadi kesalahan disaat admin memasukkan data barang.

Nama	Menghapus Data Barang
Aktor	Admin
Pre-Kondisi	Admin berhasil masuk ke halaman utama
Pra-Kondisi	Admin berhasil menghapus data barang, halaman barang
SKENARIO NORMAL “Menghapus Data Barang”	
Aktor	Sistem

1. Menekan menu barang	
	2. Menampilkan data barang: a. ID b. Nama Pengirim c. Nama Penerima d. Alamat
3. Menekan tombol hapus	
	4. Menampilkan data barang: e. ID f. Nama Pengirim g. Nama Penerima h. Alamat

4. Skenario Melihat Rute Terpendek Pengiriman Barang

Skenario melihat rute terpendek pengiriman barang merupakan fitur untuk melihat rute terpendek yang telah dihitung berdasarkan metode Bellman Ford. Fitur ini dapet diakses oleh admin.

Nama	Melihat rute terpendek pengiriman barang
Aktor	Admin, Kurir
Pre-Kondisi	Aktor sudah berhasil masuk, halaman utama
Pra-Kondisi	Aktor sudah berhasil melihat rute terpendek pengiriman barang, halaman rute terpendek
SKENARIO NORMAL	

“Melihat Rute Terpendek Pengiriman Barang”	
Aktor	Sistem
1. Menekan menu pembagian	
	2. Menampilkan data pembagian barang: a. No b. Tanggal
3. Menekan tombol detail pada sebelah kanan tanggal pengiriman yang ingin dicari	
	4. Menampilkan tabel data kurir a. No b. Nama kurir c. Kecamatan
5. Menekan tombol pekerjaan pada sebelah kanan nama kurir	
	6. .Menampilkan tabel daftar data barang disertai dengan akumulasi jarak a. No b. Kurir c. Menuju d. Jarak
7. Menekan nama paling atas pada kolom menuju	
	8.Menampilkan rute terpendek
SCENARIO ALTERNATIF	

“TIdak memiliki Antrian Pekerjaan”	
Aktor	Sistem
5.Menekan tombol pekerjaan pada sebelah kanan nama kurir	
	6. Menampilkan halaman eror
SCENARIO ALTERNATIF “TIdak Memiliki Detail Pekerjaan”	
Aktor	Sistem
3. Menekan tombol detail pada sebelah kanan tanggal pengiriman yang ingin dicari	
	6. Menampilkan halaman eror

5. Skenario Memasukkan Data Kurir

Skenario memasukkan data kurir merupakan salah satu menu yang digunakan oleh admin untuk memasukkan data kurir yang akan dikirim ke dalam sistem. Menu ini hanya dapat dilakukan oleh admin.

Nama	Memasukkan Data Kurir
Aktor	Admin
Pre-Kondisi	Aktor sudah berhasil masuk, halaman utama
Pra-Kondisi	Aktor sudah berhasil memasukkan data kurir, halaman kurir
SKENARIO NORMAL “Memasukkan Data Kurir”	

Aktor	Sistem
1. Menekan menu kurir	
	2. Menampilkan data kurir ID Username Nama lengkap Alamat Kecamatan
3. Menekan tombol tambah kurir	
	4. Menampilkan formulir pengisian data kurir : a. Username, varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)
5. Mengisi formulir data kurir	
6. Menekan tombol simpan	
	7. Menampilkan halaman data kurir
SCENARIO ALTERNATIF “Tipe data salah”	
Aktor	Sistem
1. Mengisi formulir data kurir	
2. Menekan tombol simpan	

	3. Menampilkan formulir pengisian data a. kurir : b. Username, varchar (50) c. Password, varchar (50) d. Nama lengkap, varchar (50) e. Alamat, varchar (150) f. Penempatan Kecamatan, int (11)
SCENARIO ALTERNATIF “Salah satu kolom tidak diisi”	
Aktor	Sistem
3. Mengisi formulir data kurir	
4.Menekan tombol simpan	
	5. Menampilkan formulir pengisian data kurir : a. Username, varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)

4.4.3.6 Skenario Menghapus Data Kurir

Skenario menghapus data barang merupakan menu yang digunakan oleh admin untuk menghapus data barang jika terjadi kesalahan disaat admin memasukkan data barang.

Nama	Menghapus Data Kurir

Aktor	Admin
Pre-Kondisi	Admin berhasil masuk ke halaman utama
Pra-Kondisi	Admin berhasil menghapus data kurir halaman kurir
SKENARIO NORMAL “Menghapus Data Kurir ”	
Aktor	Sistem
1. Menekan menu kurir	
	2. Menampilkan data kurir: a. ID b. Username c. Nama Lengkap d. Alamat e. Kecamatan
3. Menekan tombol hapus	
	4. Menampilkan data kurir: a. ID b. Username c. Nama Lengkap d. Alamat e. Kecamatan

6. *Use Case* Skenario Melihat Data Kurir

Skenario melihat data kurir merupakan salah satu fitur yang disediakan oleh sistem untuk menampilkan seluruh nama pegawai kurir yang dimiliki. Fitur ini dapat diakses oleh admin saja.

Nama	Melihat data kurir
Aktor	Admin
Pre-Kondisi	Admin berhasil masuk, halaman utama
Pra-Kondisi	Admin berhasil melihat data kurir, halaman kurir
SKENARIO NORMAL "Melihat data kurir"	
Aktor	Sistem
1. Menekan menu kurir	
	2. Menampilkan data kurir yang berisi : a) ID b) Username c) Nama Lengkap d) Alamat

7. Melihat Daftar Pengiriman Barang

Skenario melihat daftar pengiriman barang merupakan salah satu fitur yang disediakan oleh sistem untuk menampilkan seluruh daftar nama barang yang akan dikirimkan. Fitur ini dapat diakses oleh admin saja.

Nama	Melihat daftar pengiriman barang
Aktor	Admin

Pre-Kondisi	Admin berhasil masuk, halaman utama
Pra-Kondisi	Admin berhasil melihat daftar pengiriman barang, halaman barang
SKENARIO NORMAL "Melihat daftar pengiriman barang"	
Aktor	Sistem
1. Menekan menu barang	
	2. Menampilkan tabel pengiriman barang yang berisi: a) ID b) Nama Pengirim c) Nama Penerima d) Alamat

8. Melihat Daftar Pembagian Barang

Skenario melihat daftar pengiriman barang merupakan salah satu fitur yang disediakan oleh sistem untuk menampilkan seluruh daftar nama barang yang akan dikirimkan. Fitur ini dapat diakses oleh admin saja.

Nama	Melihat daftar pembagian barang
Aktor	Kurir
Pre-Kondisi	Kurir berhasil masuk, halaman utama
Pra-Kondisi	Kurir berhasil melihat daftar pembagian barang, halaman hasil pembagian barang
SKENARIO NORMAL "Melihat daftar pembagian barang"	

Aktor	Sistem
1. Menekan tombol detail	
	2. Menampilkan tabel hasil pembahian barang yang berisi: a) No b) Kurir c) Menuju d) Jarak

9. Mencari Data Tanggal Pengiriman Barang

Skenario mencari data tanggal pengiriman barang merupakan fitur yang disediakan sistem untuk mempermudah mencarian pengiriman barang pada tanggal tertentu. Fitur ini dapat diakses oleh dua aktor yaitu admin dan kurir.

Nama	Mencari data tanggal pengiriman barang
Aktor	Admin, Kurir
Pre-Kondisi	Admin berhasil masuk ke halaman utama
Pra-Kondisi	Admin berhasil melakukan pencarian, halaman pembagian
SKENARIO NORMAL “Mencari data tanggal pengiriman barang”	
Aktor	Sistem
1. Menekan menu pembagian	
	2. Menampilkan tabel pembagian barang: a. No

	b. Tanggal
3. Mengisi kolom *search*	
	4. Menampilkan hasil pencarian sesuai dengan tanggal yang diisikan pada kolom *search*: a) No b) Tanggal

10. Keluar

Skenario keluar merupakan fitur yang berfungsi untuk keluar dari sistem. Fitur ini dapat diakses oeh dua actor yaitu admin dan kurir.

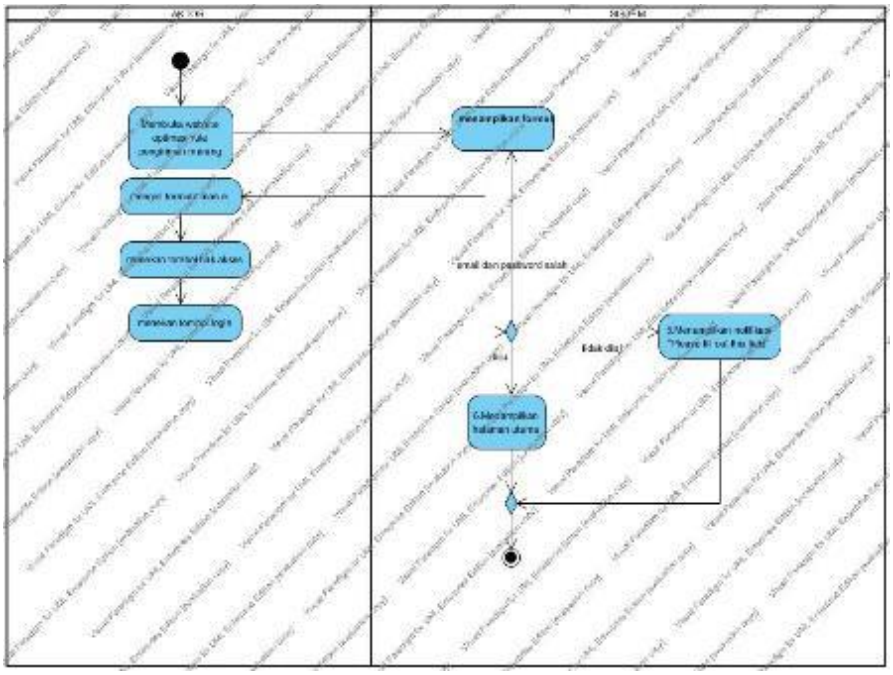
Nama	Keluar
Aktor	Admin, Kurir
Pre-Kondisi	Aktor berhasil masuk, halaman utama
Pra-Kondisi	Aktor berhasil keluar, halaman masuk
SKENARIO NORMAL “Keluar”	
Aktor	Sistem
1. Menekan tombol logout	
	2. Menampilkan formulir masuk yang berisi: b. E-mail Addres, varchar (255) c. Password, varchar (255)

4.4.3 *Activity Diagram*

Activity Diagram merupakan dokumentasi desain yang menggambarkan aliran aktivitas dalam Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford yang akan di bangun. Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford ini memiliki sebelas *activity* diagram yaitu sebagai berikut:

1 *Activity Diagram* Masuk

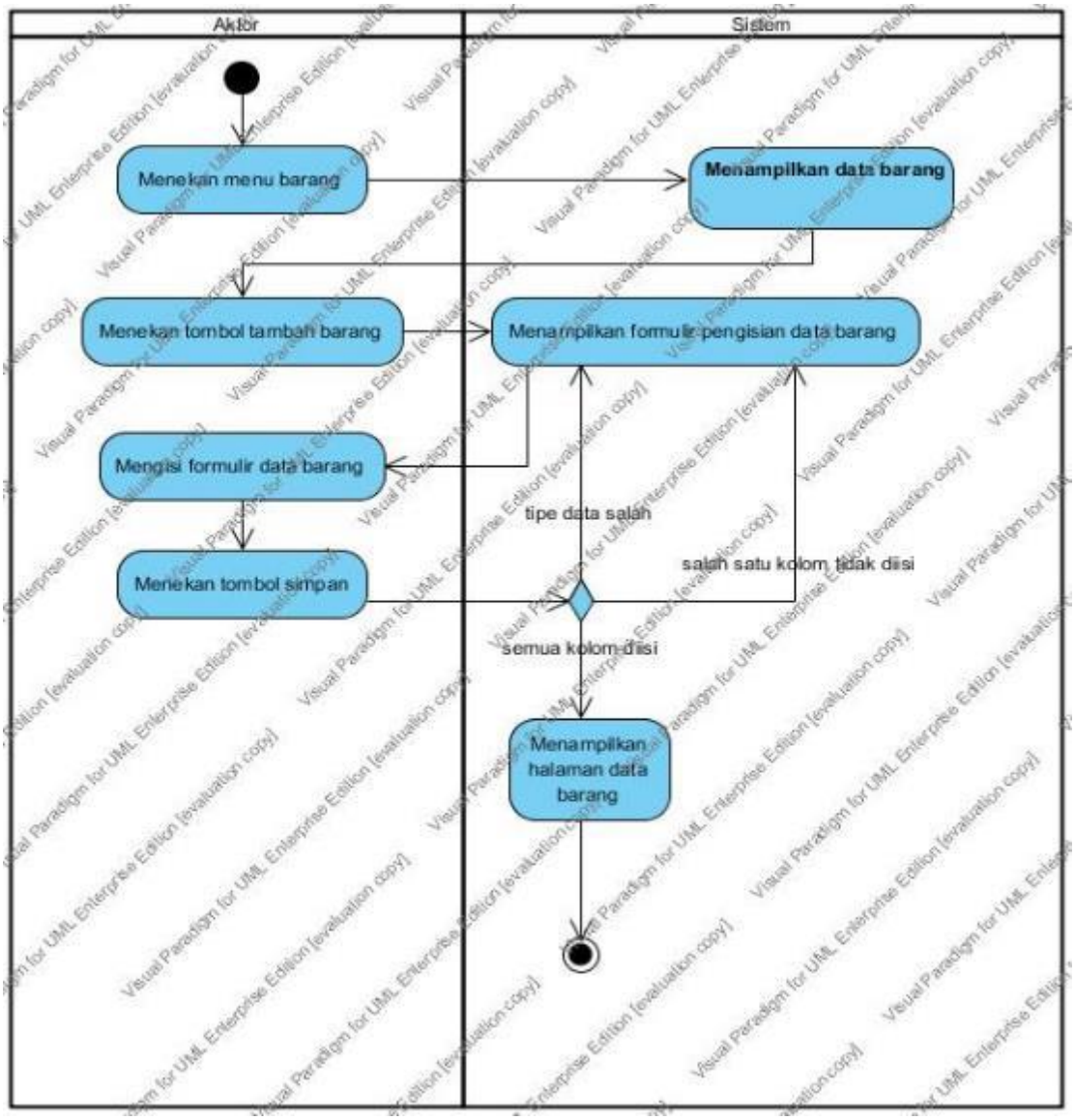
Activity diagram masuk dilakukan oleh aktor sistem. *Activity diagram* masuk menjelaskan tentang bagaimana sistem dapat menjalankan fungsi melakukan autentifikasi hak akses aktor dalam menggunakan sistem.



Gambar 4. 3 Activity Diagram Masuk

2 *Activity Diagram* Memasukkan Data Barang

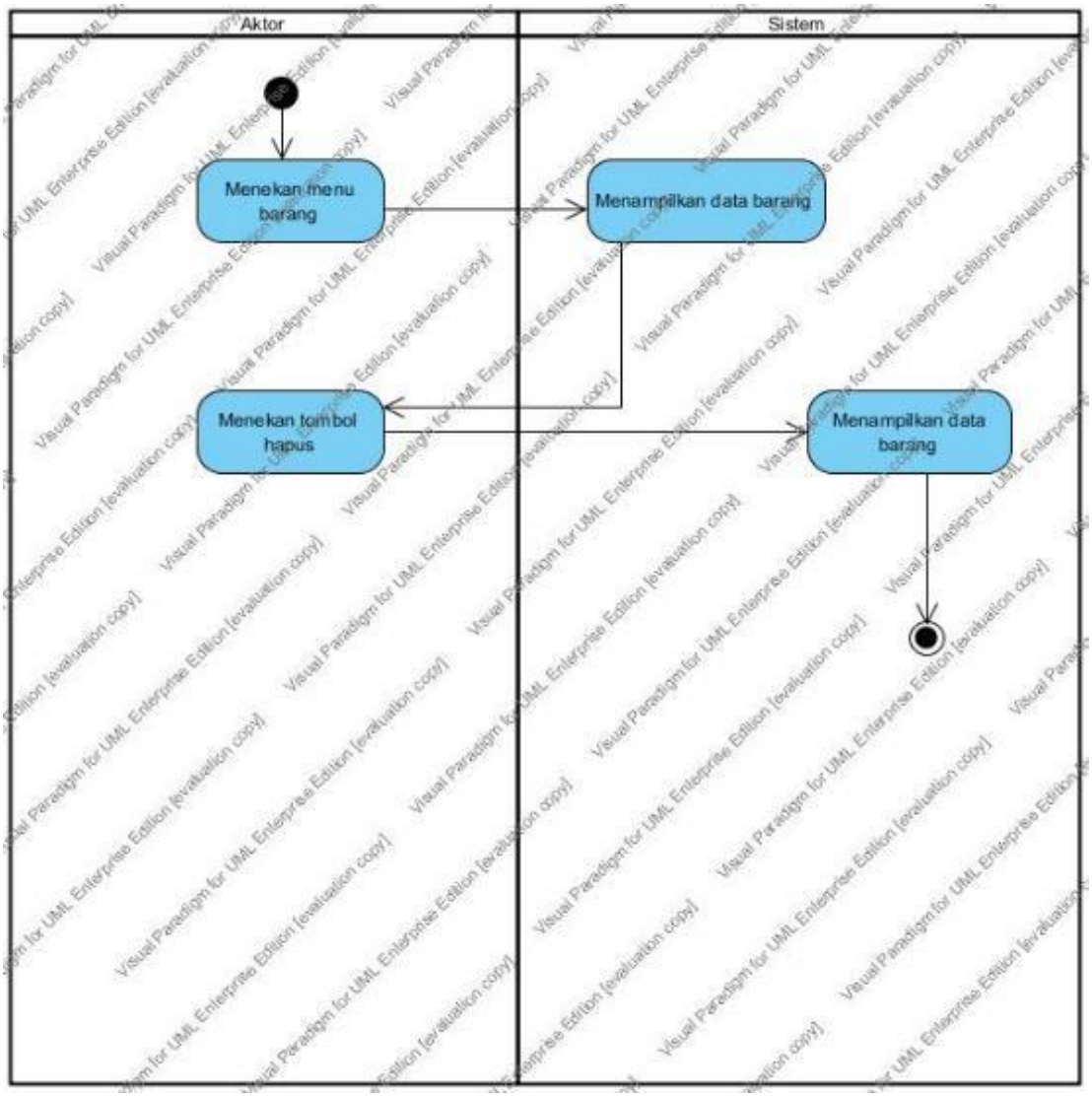
Activity diagram memasukkan data barang menjelaskan alur atau tata cara admin untuk memasukkan data barang ke dalam sistem.



Gambar 4. 4 Activity Diagram Memasukkan Data Barang

3 *Activity Diagram* Menghapus Data Barang

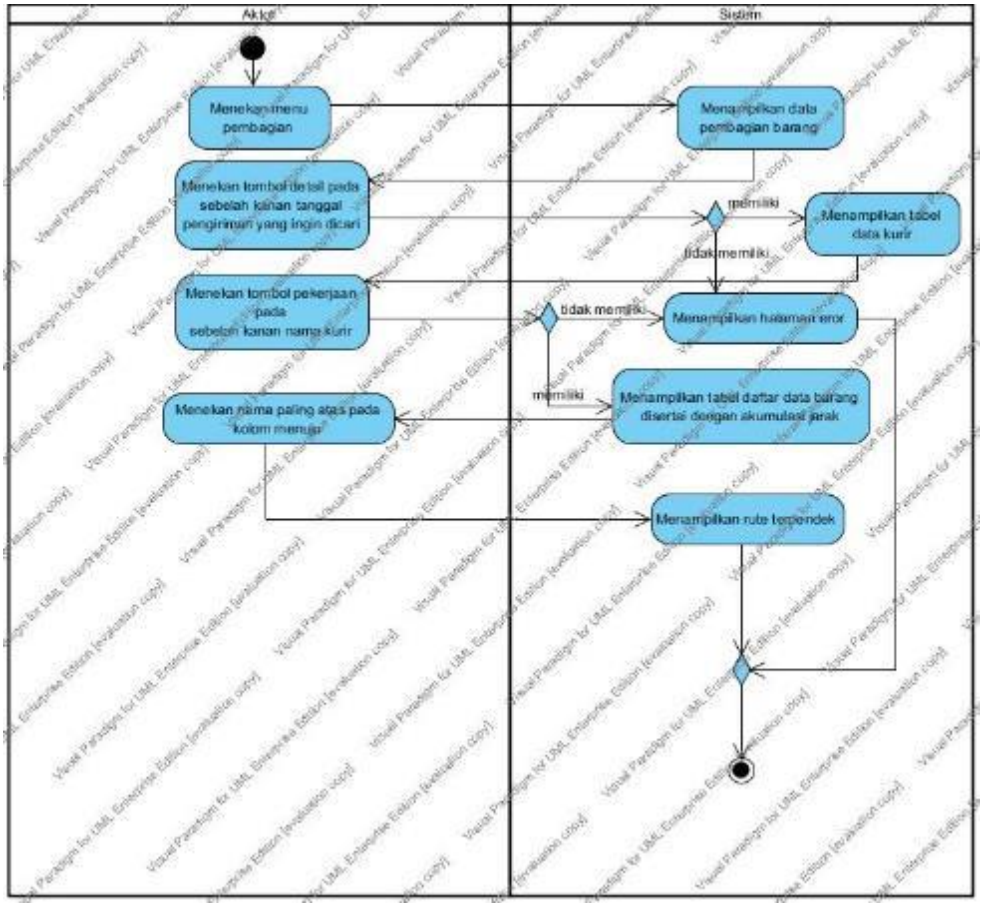
Activity diagram menghapus data barang merupakan penjelasan alur atau tata cara admin dalam menghapus data barang.



Gambar 4. 5 Activity Diagram Menghapus Data Barang

4 *Activity Diagram* Melihat Rute Terpendek Pengiriman Barang

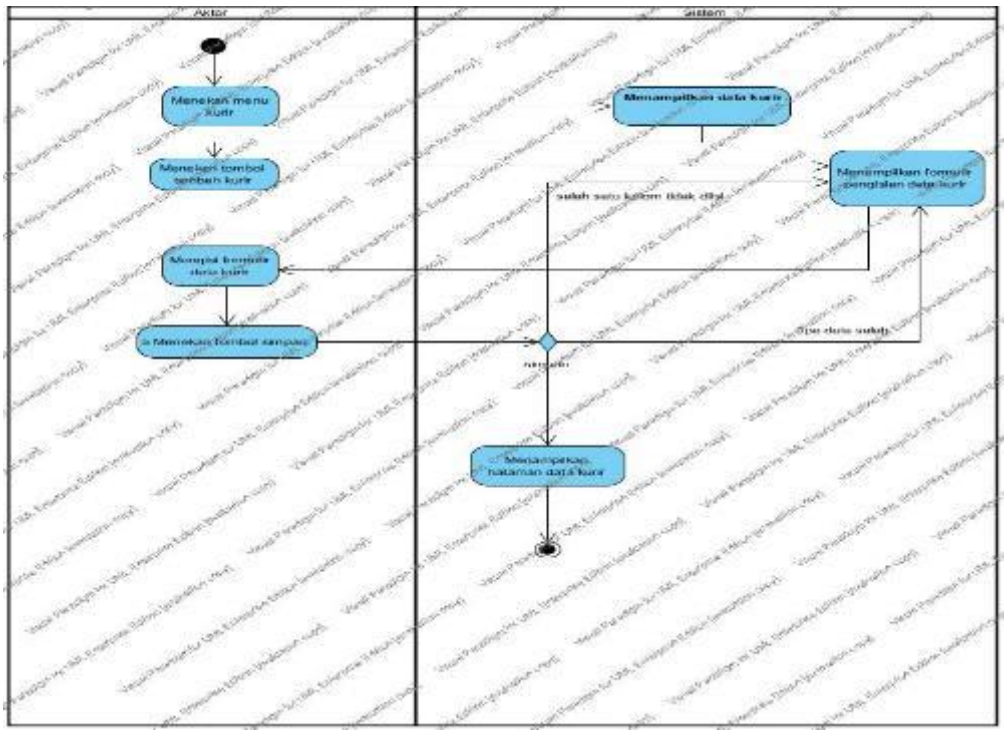
Activity diagram melihat rute terpendek pengiriman barang merupakan alur atau tata cara actor (admin dan kurir) melihat rute terpendek pengiriman barang yang dilakukan.



Gambar 4. 6 Activity Diagram Melihat Rute Terpendek Pengiriman Barang

5 *Activity Diagram* Memasukkan Data Kurir

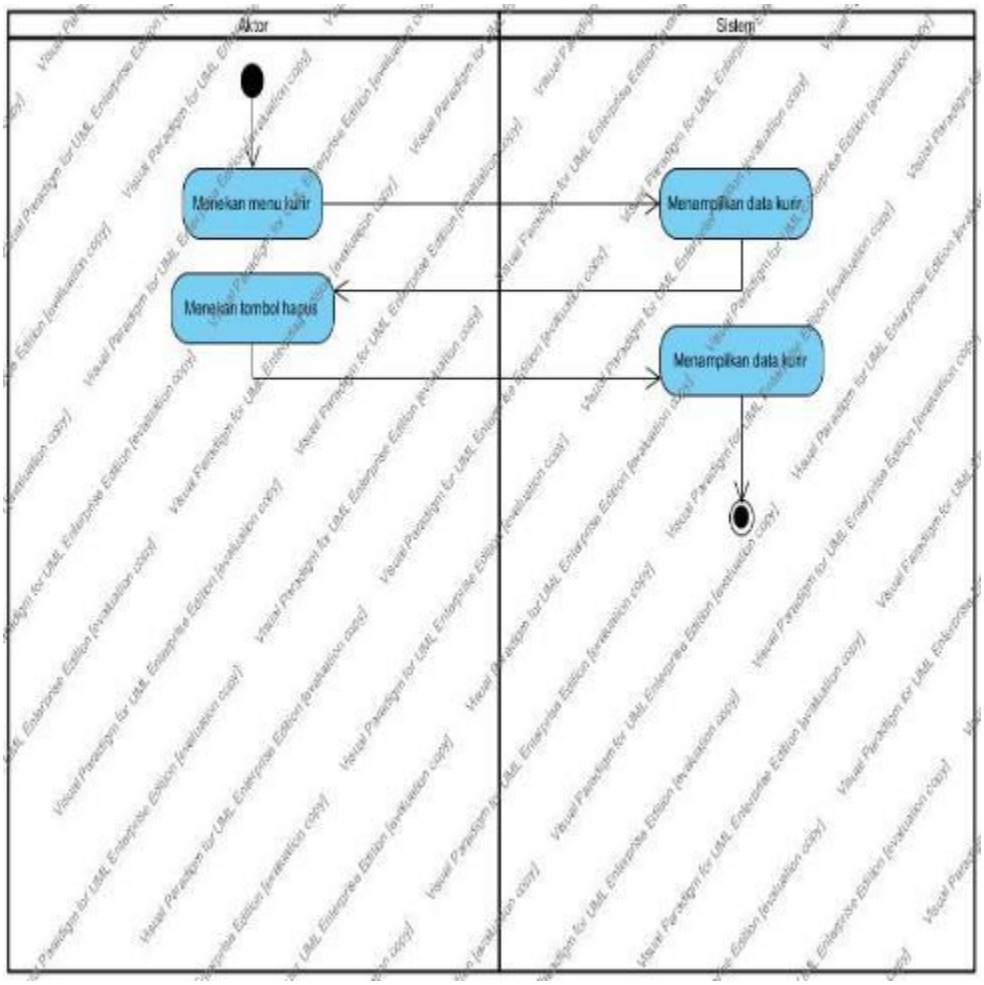
Activity diagram memasukkan data kurir menjelaskan alur atau tata cara admin untuk memasukkan data kurir ke dalam sistem.



Gambar 4. 7 Activity Diagram Memasukkan Data Kurir

6 *Activity Diagram* Menghapus Data Kurir

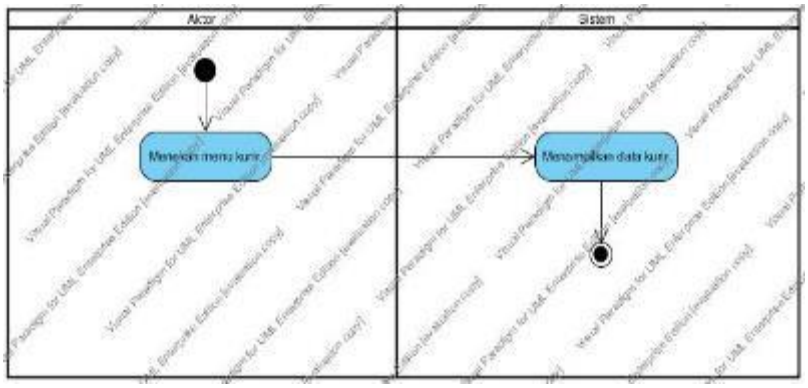
Activity diagram menghapus data kurir merupakan penjelasan alur atau tata cara admin dalam menghapus data kurir.



Gambar 4. 8 Activity Diagram Menghapus Data Kurir

7 *Activity Diagram* Melihat Data Kurir

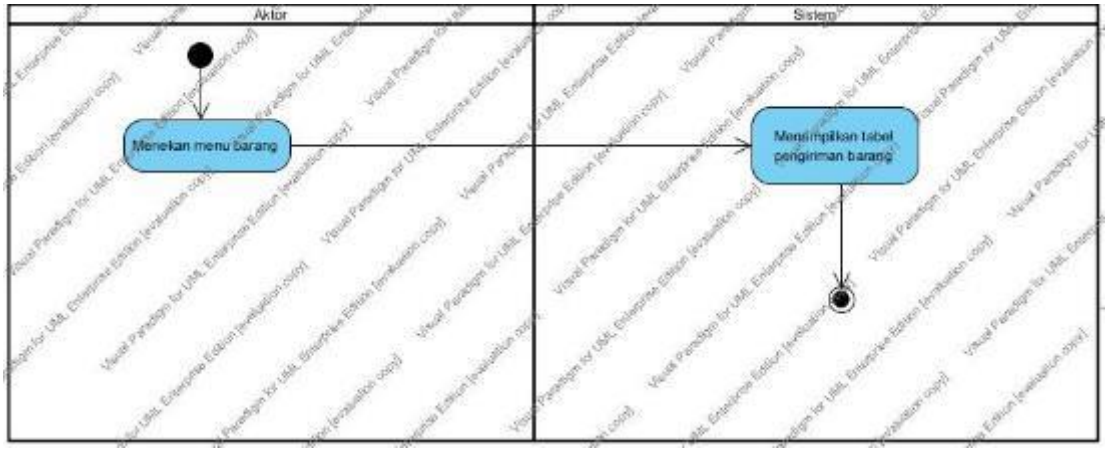
Activity diagram melihat data kurir merupakan alur atau tata cara yang menjelaskan tentang cara melihat data kurir yang dilakukan oleh admin.



Gambar 4. 9 Activity Diagram Melihat Data Kurir

8 *Activity Diagram* Melihat Daftar Pengiriman Barang

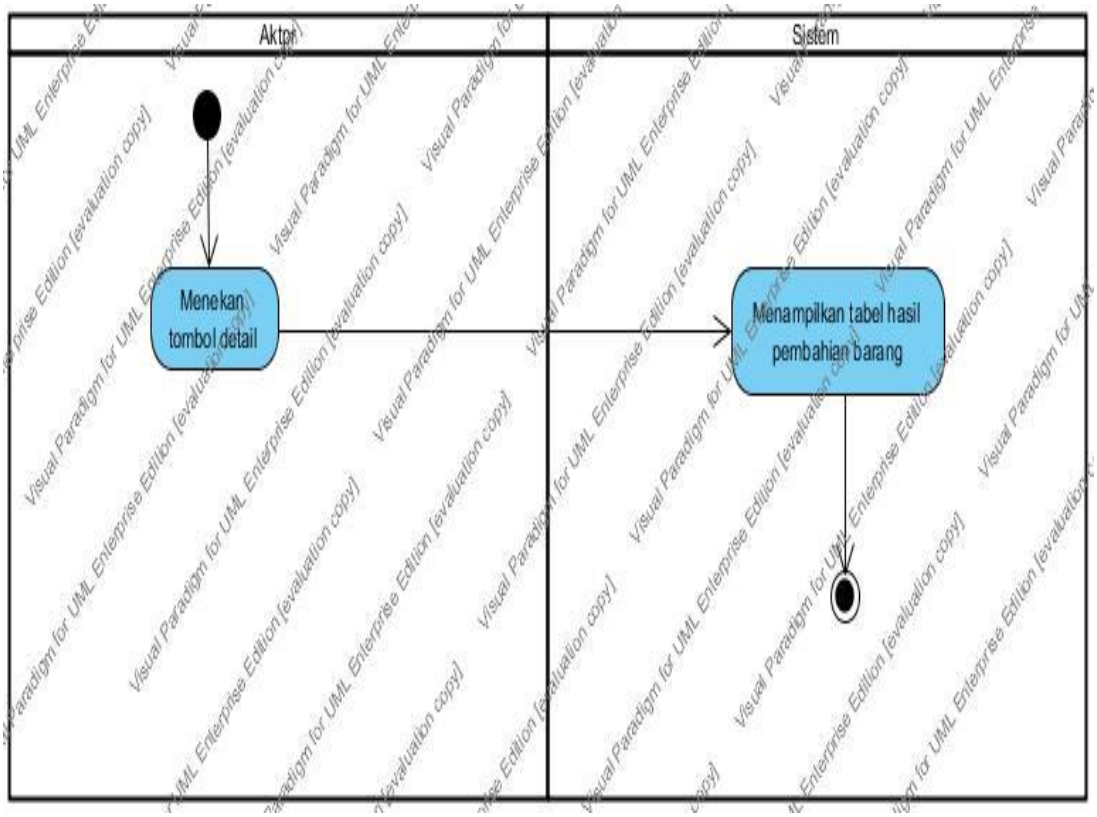
Activity diagram melihat daftar pengiriman barang merupakan alur atau tata cara admin untuk melihat daftar pengiriman barang yang akan dikirim.



Gambar 4. 10 Activity Diagram Melihat Daftar Pengiriman Barang

9 *Activity Diagram* Melihat Daftar Pembagian Barang

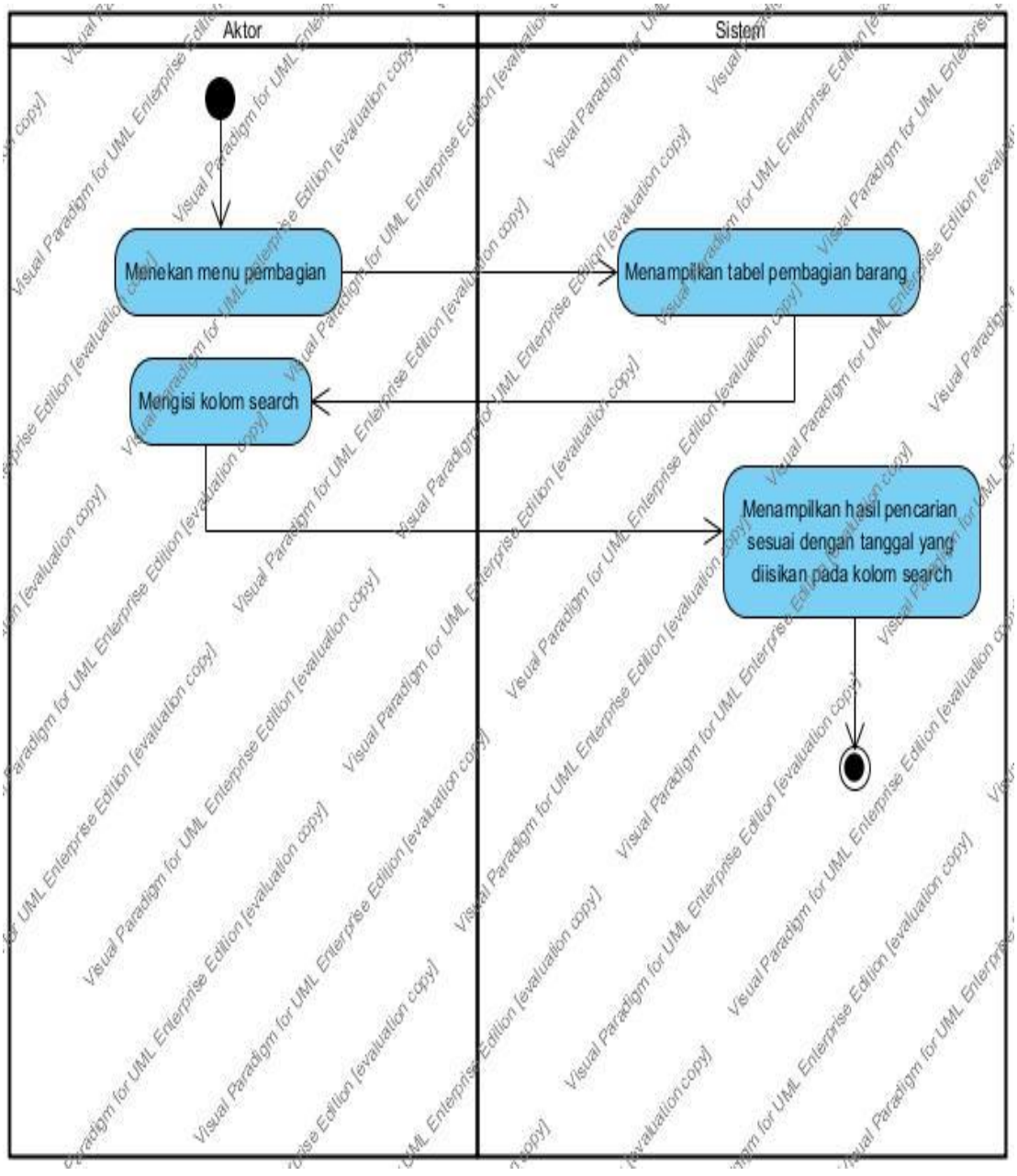
Activity diagram melihat daftar pembagian barang merupakan alur atau tata cara sistem untuk menampilkan daftar pembagian barang yang dilakukan oleh kurir.



Gambar 4. 11 Activity Diagram Melihat Daftar Pembagian Barang

10 *Activity Diagram* Mencari Data Tanggal Pengiriman Barang

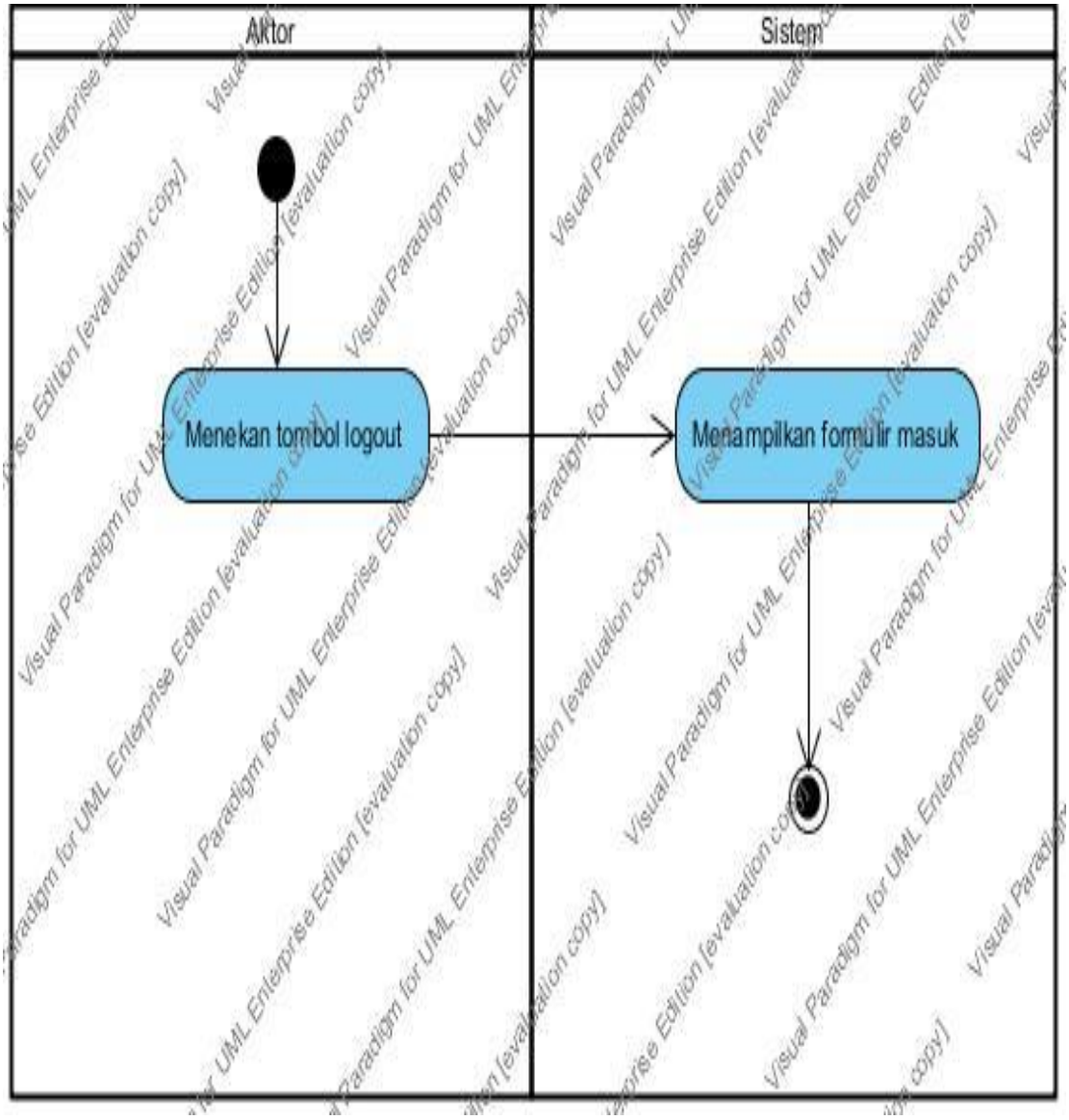
Activity diagram mencari data tanggal pengiriman barang merupakan alur atau tata cara sistem untuk mencari data tanggal pengoroman barang yang dilakukan oleh dua actor yaitu admin dan kurir.



Gambar 4. 12 Activity Diagram Mencari Data Tanggal Pengiriman Barang

11 *Activity Diagram* Keluar

Activity diagram keluar merupakan alur atau tata cara actor (admin dan kurir) untuk keluar dari sistem.



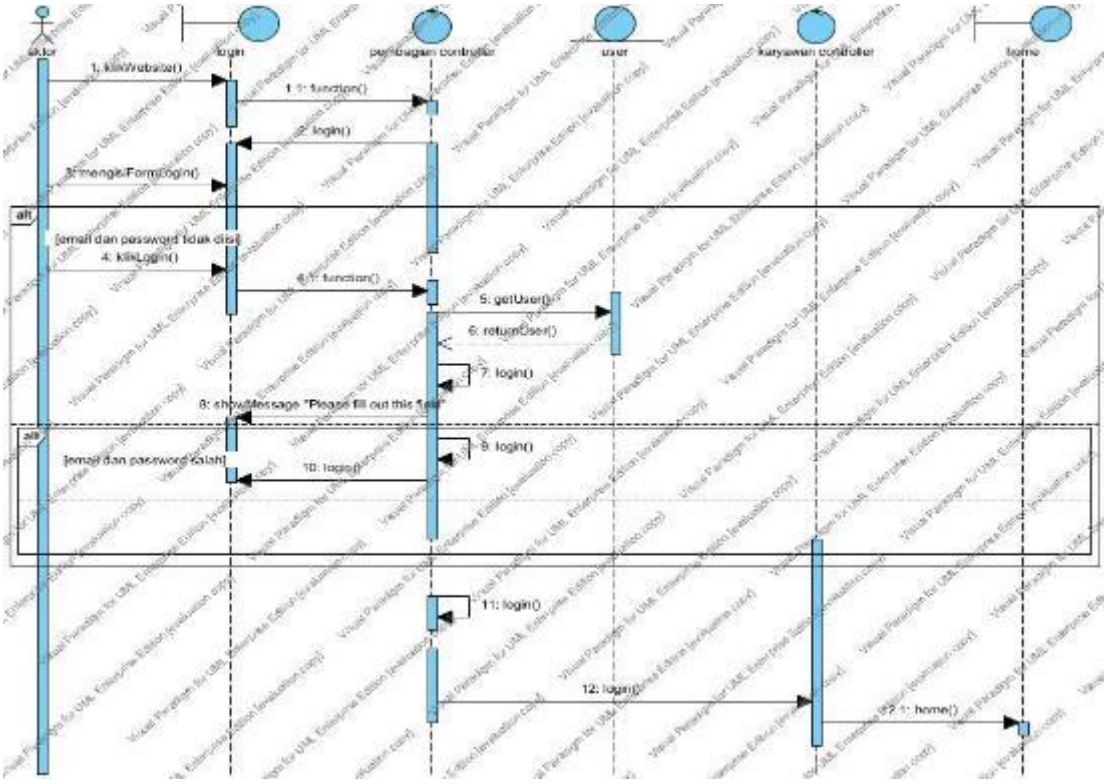
Gambar 4. 13 Activity Diagram Keluar

4.4.5 *Sequence* Diagram

Sequence diagram adalah dokumentasi suatu diagram yang menampilkan urutan interaksi - interaksi antar objek di dalam sistem. *Sequence* diagram digunakan untuk menggambarkan skenario dan memodelkan aliran logika dalam sistem dengan cara *visual*. *Sequence* diagram dari Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford adalah sebagai berikut:

1 *Sequence* Diagram Masuk

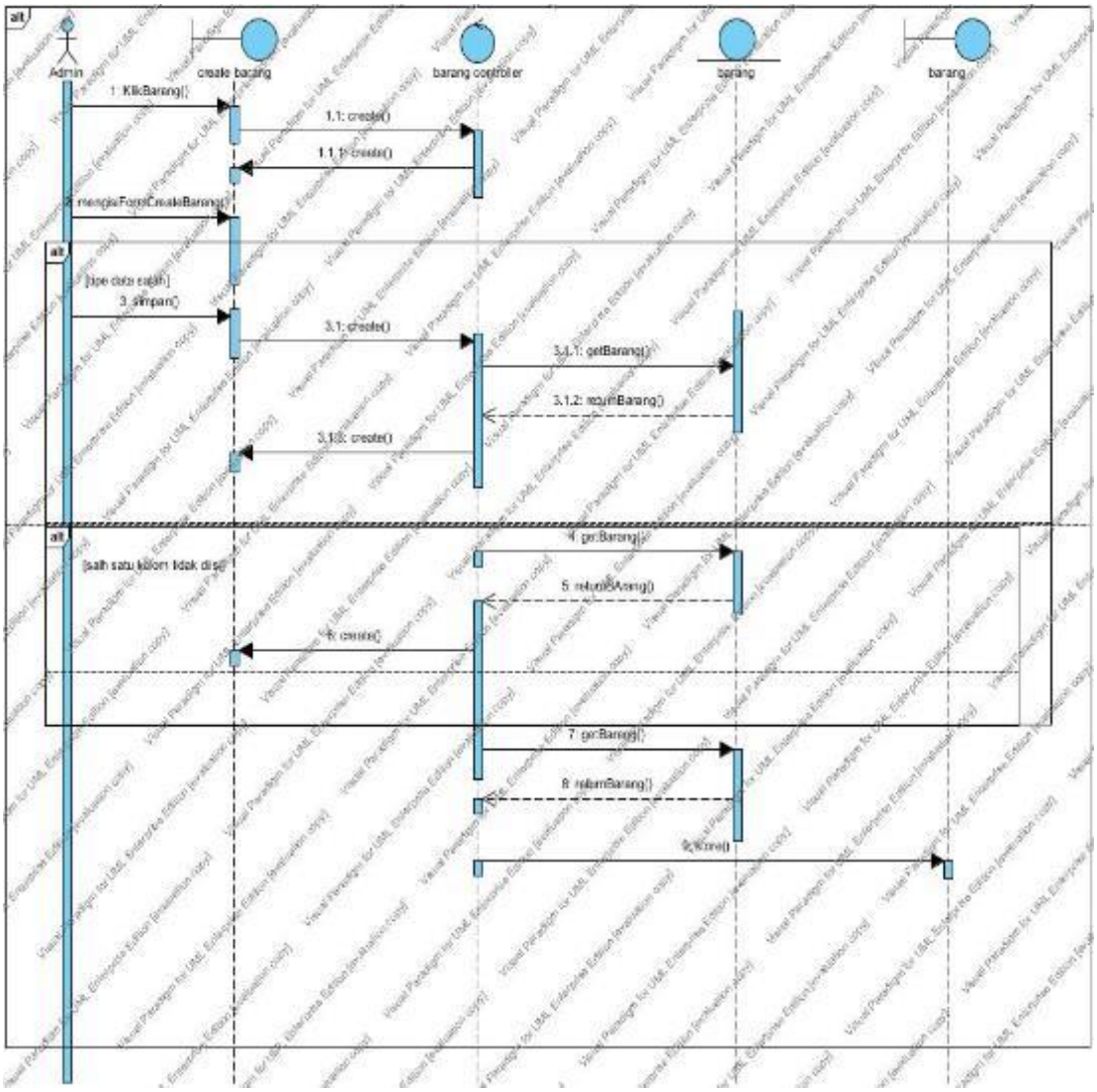
Penggambaran *sequence* diagram masuk digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* masuk yang menjelaskan dua aktor yaitu admin dan kurir untuk masuk sistem. Masing - masing *class* ditampilkan secara *visual* dengan gambar.



Gambar 4. 14 Sequence Diagram Masuk

2 *Sequence* Diagram Memasukkan Data Barang

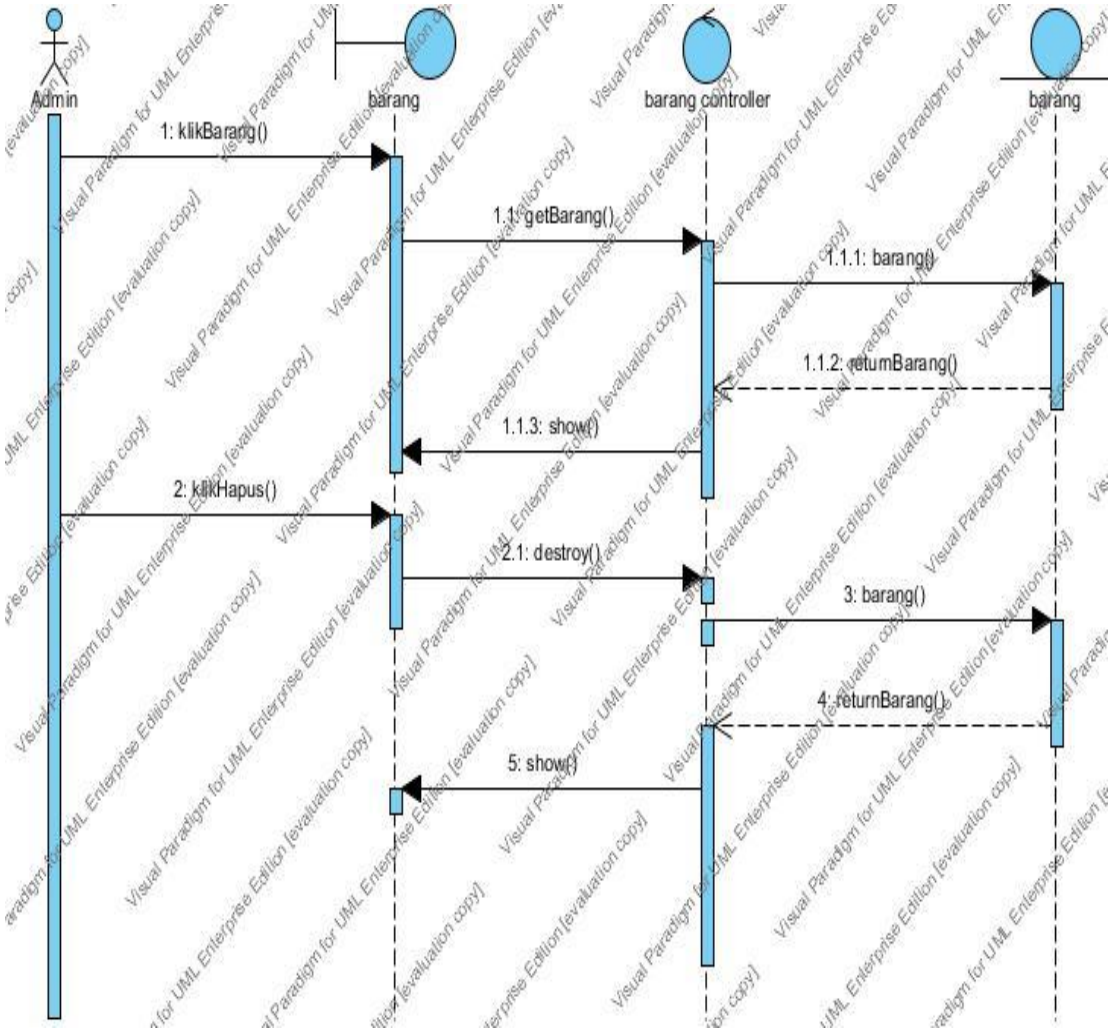
Penggambaran *sequence* diagram memasukkan data barang digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* memasukkan data barang yang dilakukan oleh admin. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 15 Sequence Diagram Memasukkan Data Barang

3 *Sequence* Diagram Menghapus Data Barang

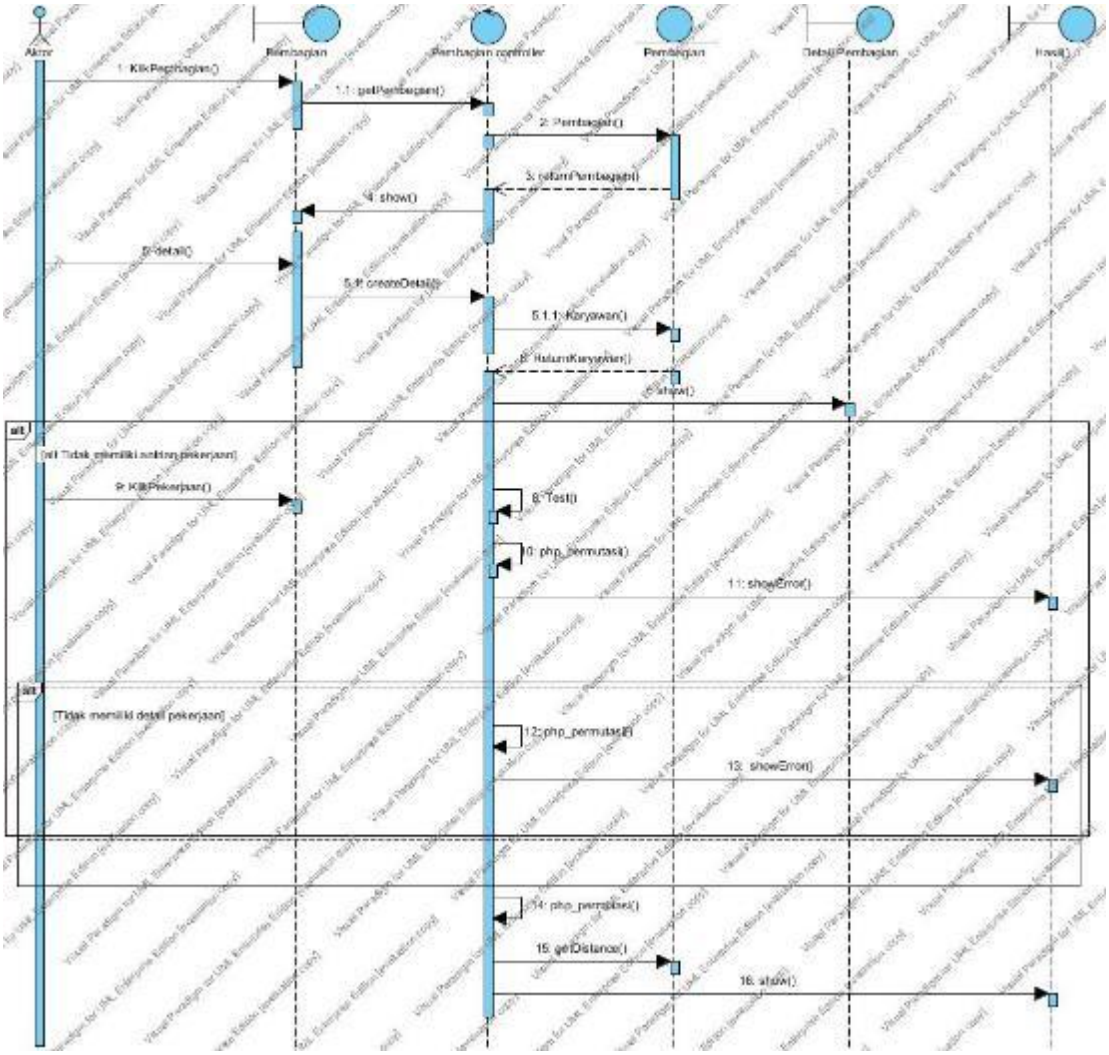
Penggambaran *sequence* diagram menghapus data barang digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* menghapus data barang yang dilakukan oleh admin. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 16 Sequence Diagram Menghapus Data Barang

4 *Sequence* Diagram Melihat Rute Terpendek Pengiriman Barang

Penggambaran *sequence* melihat rute terpendek pengiriman barang digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* melihat rute terpendek pengiriman barang yang dilakukan oleh dua actor yaitu admin dan kurir. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.

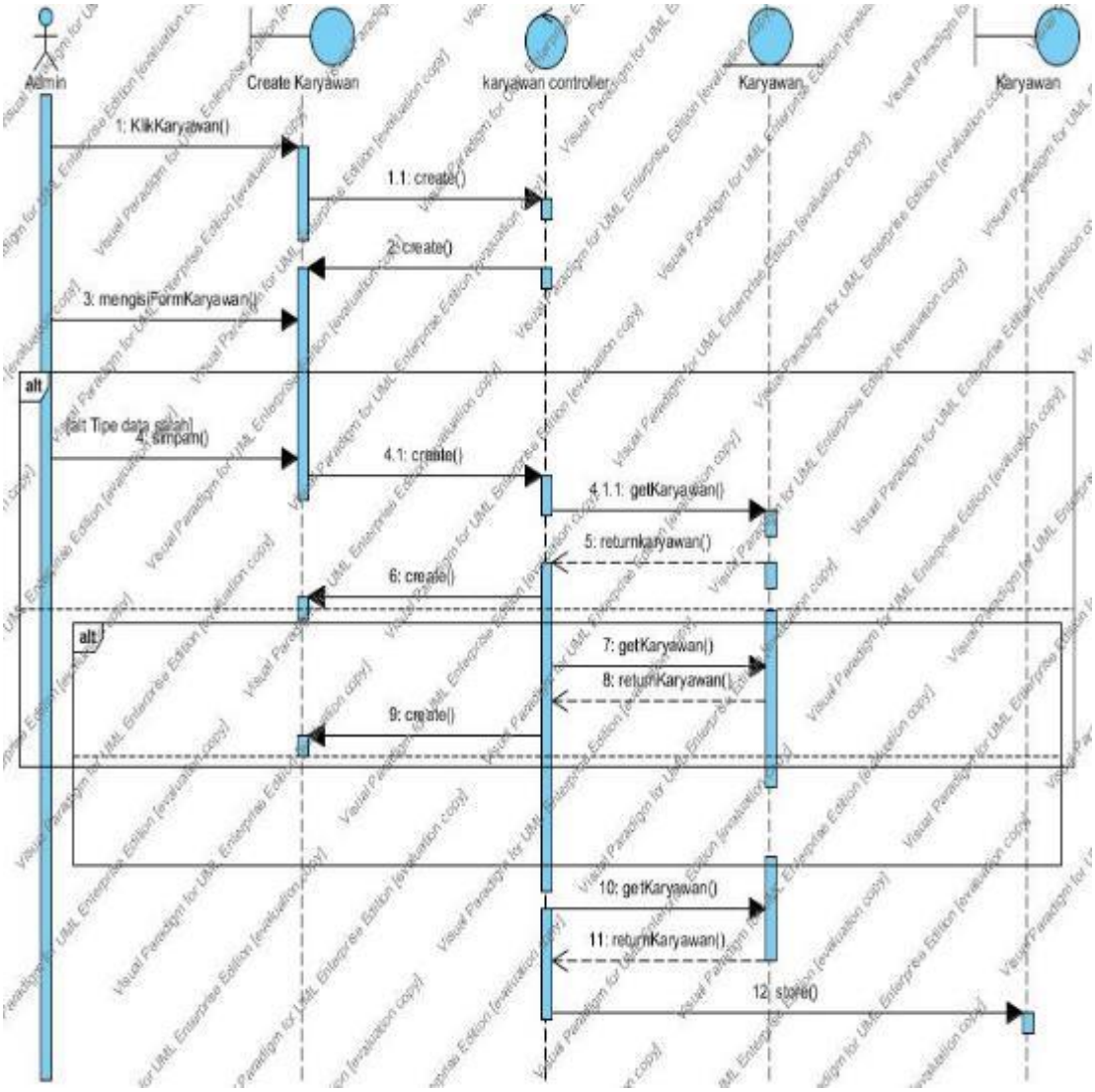


Gambar 4. 17 Sequence Diagram Melihat Rute Terpendek Pengiriman Barang

5 *Sequence* Diagram Memasukkan Data Kurir

Penggambaran *sequence* diagram memasukkan data kurir digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* memasukkan data

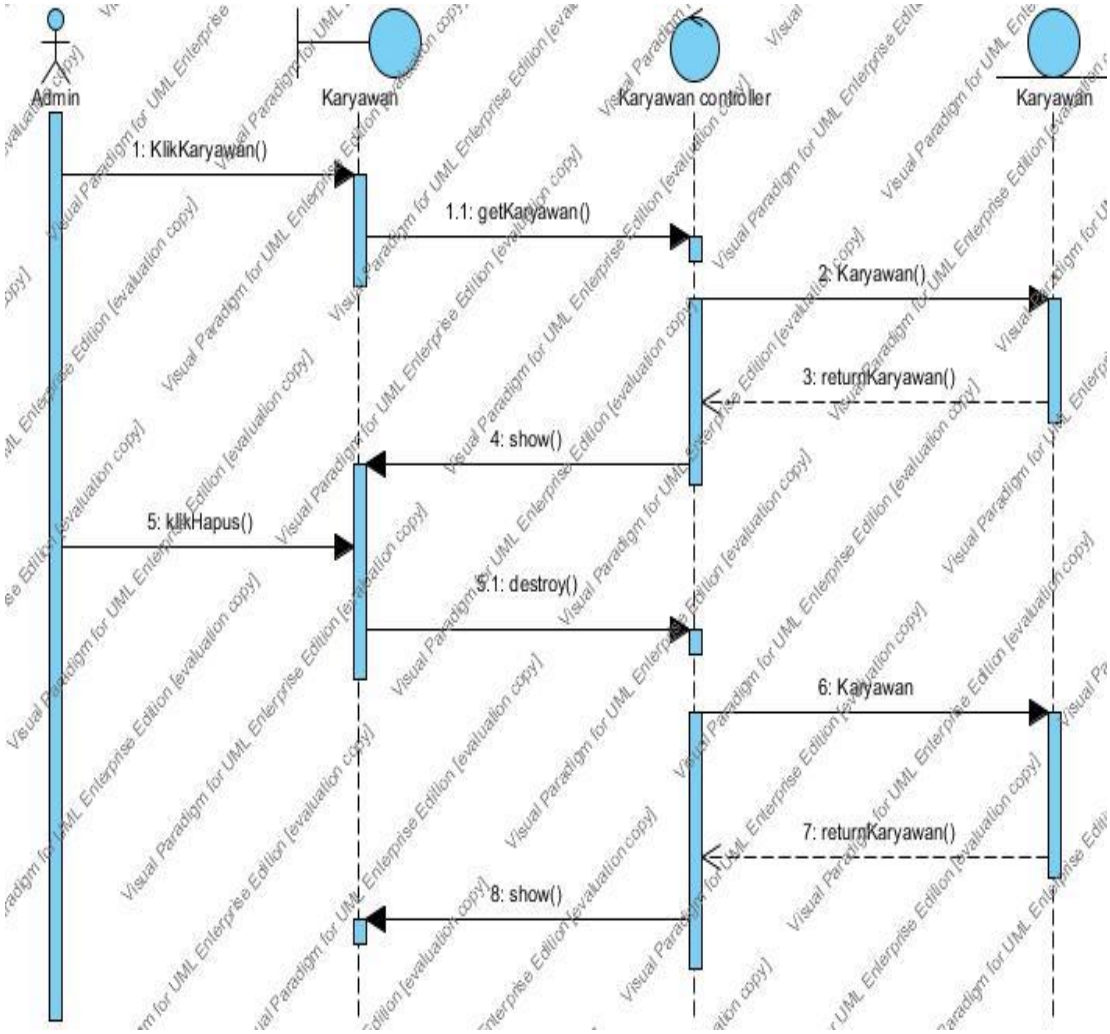
kurir yang dilakukan oleh admin. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 18 Sequence Diagram Memasukkan Data Kurir

6 *Sequence* Diagram Menghapus Data Kurir

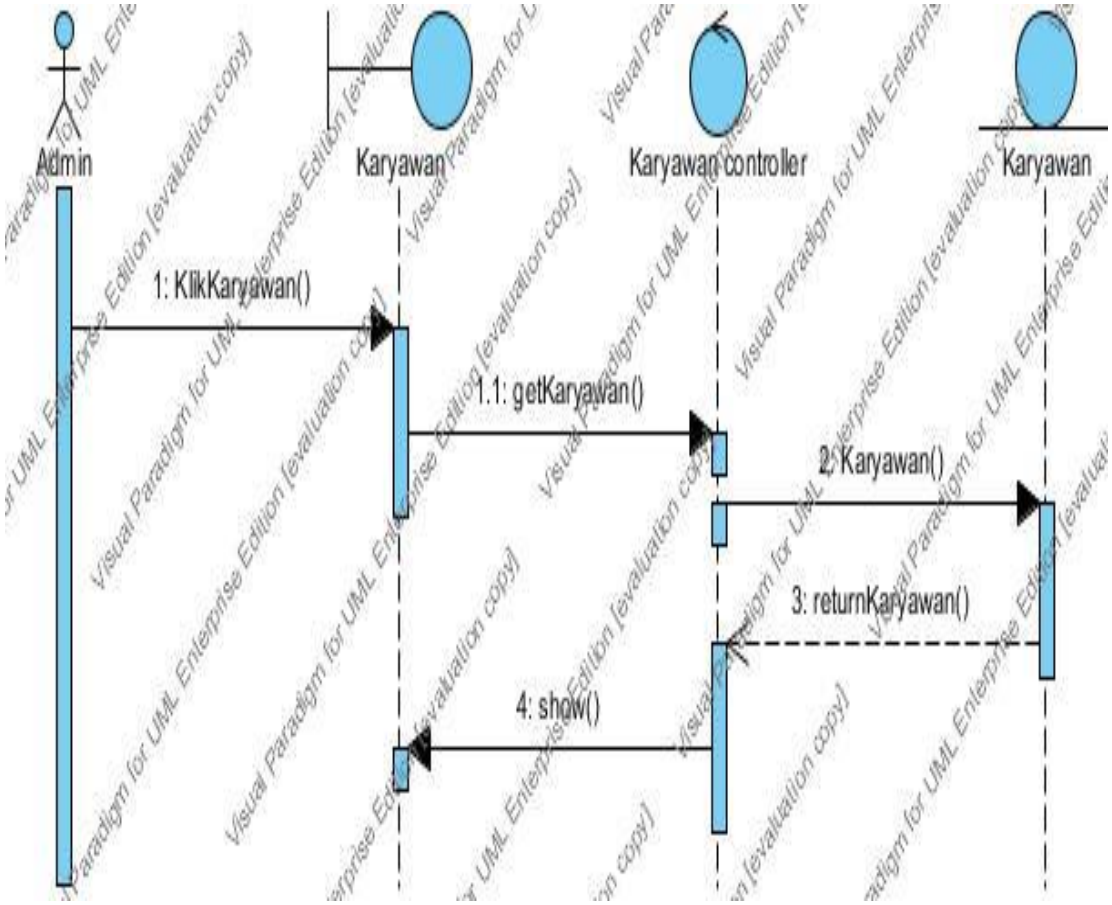
Penggambaran *sequence* diagram menghapus data kurir digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* menghapus data kurir yang dilakukan oleh admin. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 19 Sequence Diagram Menghapus Data Kurir

7 *Sequence* Diagram Melihat Data Kurir

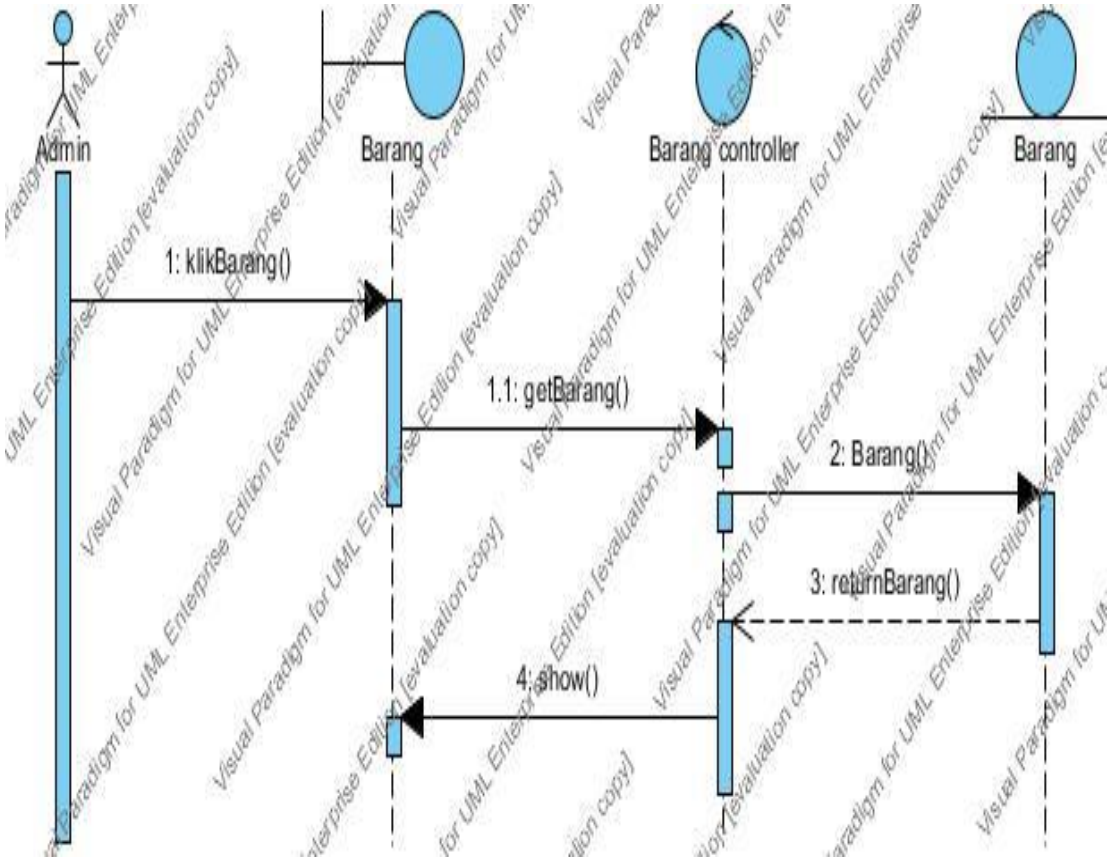
Penggambaran *sequence* diagram melihat data kurir digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* melihat data kurir yang dilakukan oleh admin. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 20 Sequence Diagram Melihat Data Kurir

8 *Sequence* Diagram Melihat Daftar Pengiriman Barang

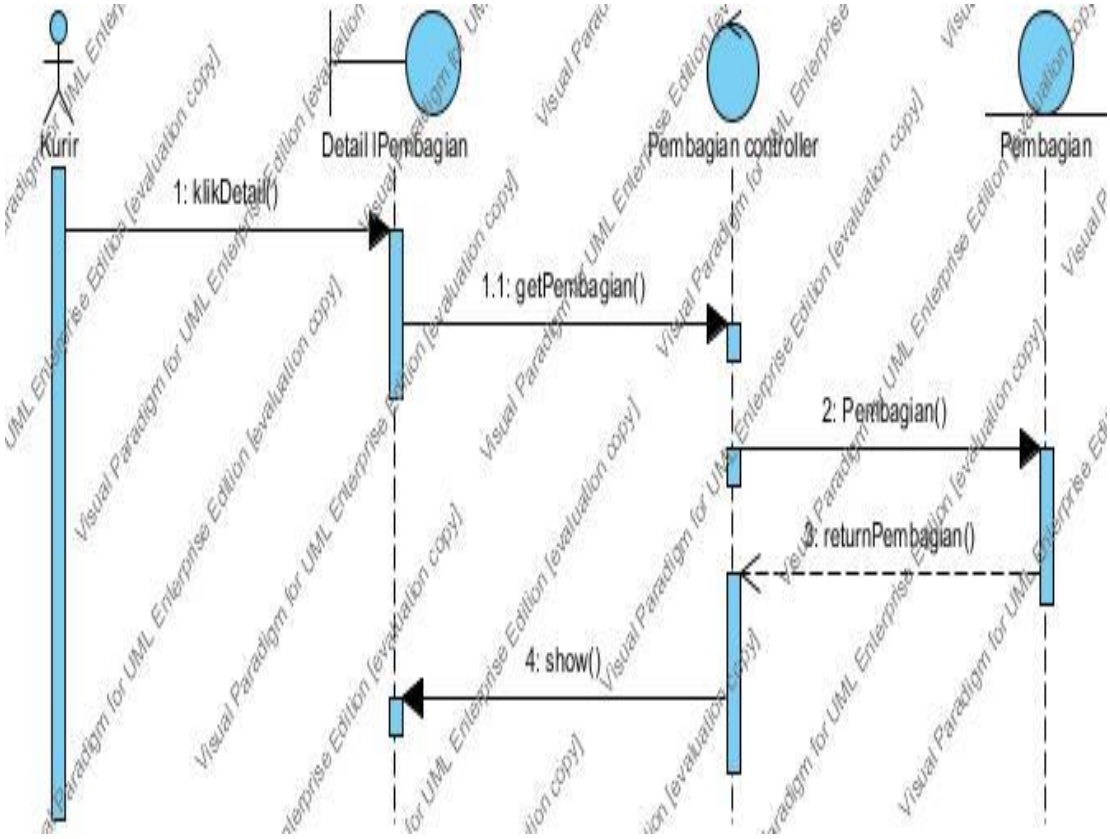
Penggambaran *sequence* diagram melihat daftar pengiriman barang digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* melihat daftar pengiriman barang dilakukan oleh admin. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar



Gambar 4. 21 Sequence Diagram Melihat Daftar Pengiriman Barang

9 Diagram Melihat Daftar Pembagian Barang

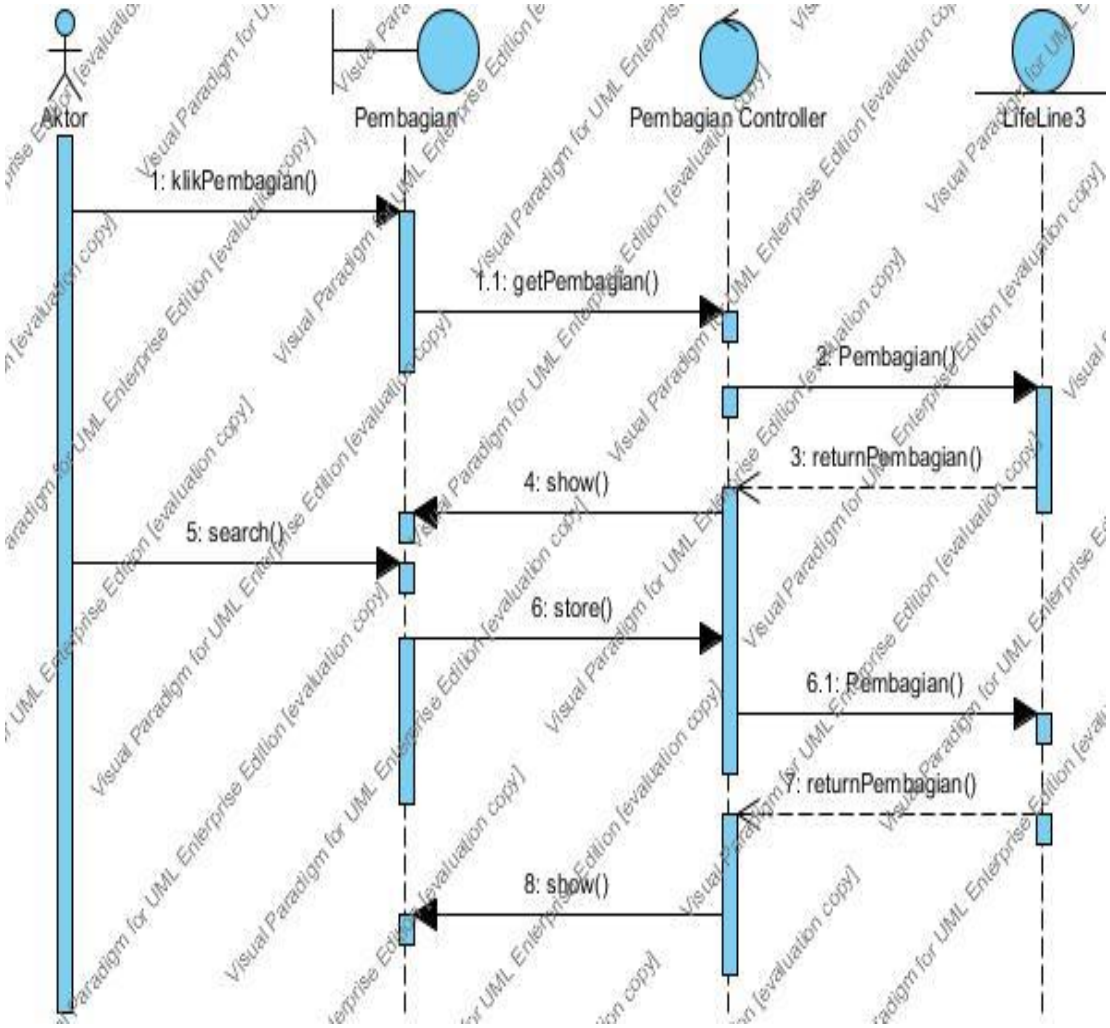
Penggambaran *sequence* diagram melihat daftar pembagian barang digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* melihat daftar pembagian barang dilakukan oleh kurir. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 22 Sequence Diagram Melihat Daftar Pembagian Barang

10 *Sequence* Diagram Mencari Data Tanggal Pengiriman Barang

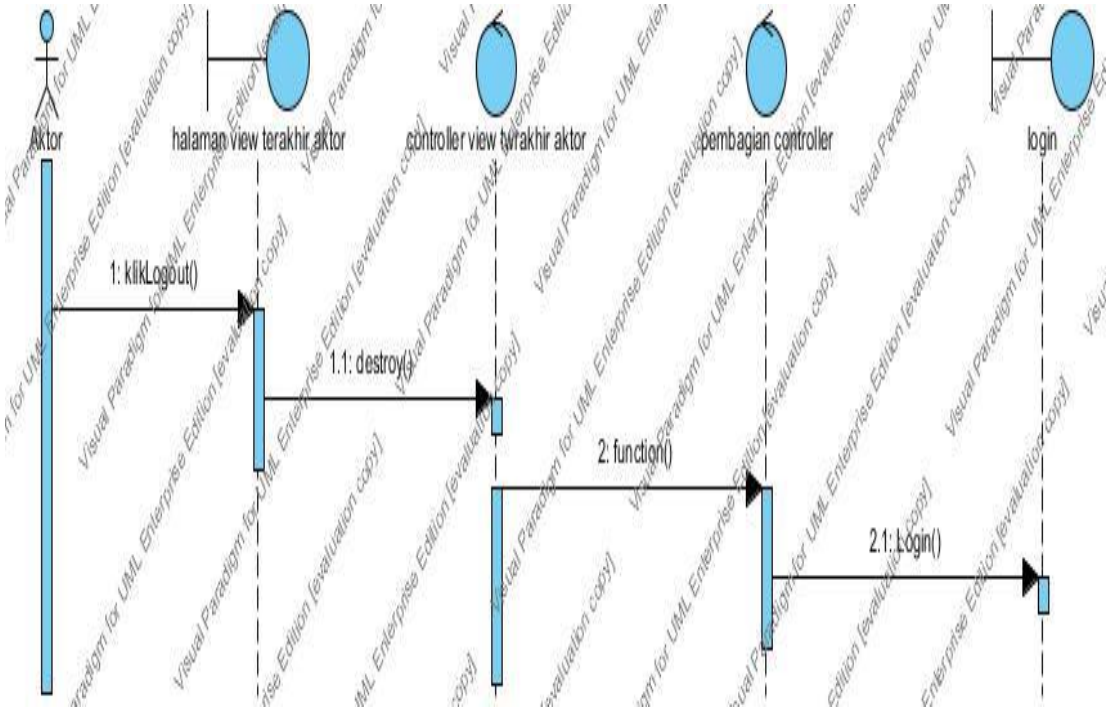
Penggambaran *sequence* diagram mencari data tanggal pengiriman barang digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* mencari data tanggal pengiriman barang dilakukan oleh dua actor yaitu admin dan kurir. Masing - masing *class* akan ditampilkan secara *visual* dengan gambar.



Gambar 4. 23 Sequence Diagram Mencari Data Tanggal Pengiriman Barang

11 *Sequence* Diagram Keluar

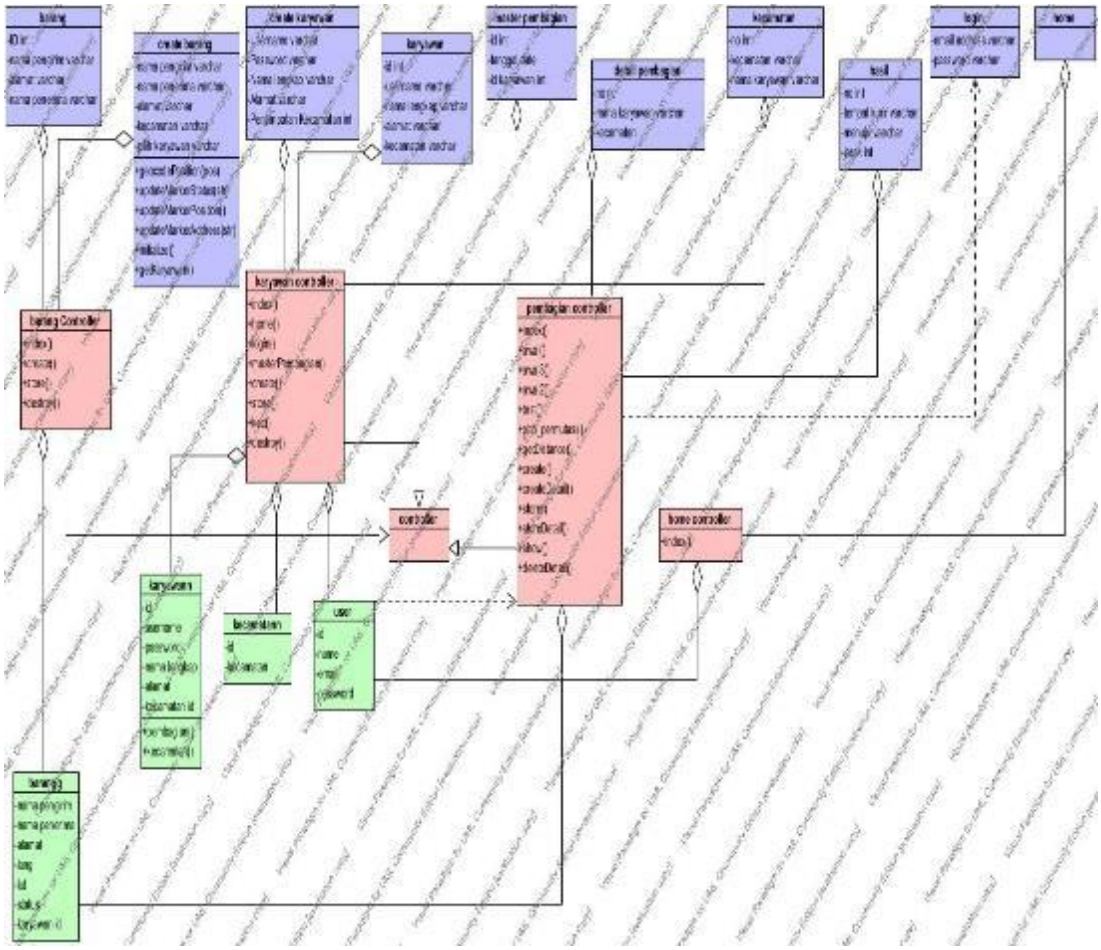
Penggambaran *sequence* diagram keluar digunakan untuk menjelaskan fungsi atau *method* yang akan dibuat pada *use case* keluar yang menjelaskan dua aktor yaitu admin dan kurir untuk keluar sistem. Masing - masing *class* ditampilkan secara *visual* dengan gambar.



Gambar 4. 24 Sequence Diagram Keluar

4.4.6 *Class* Diagram

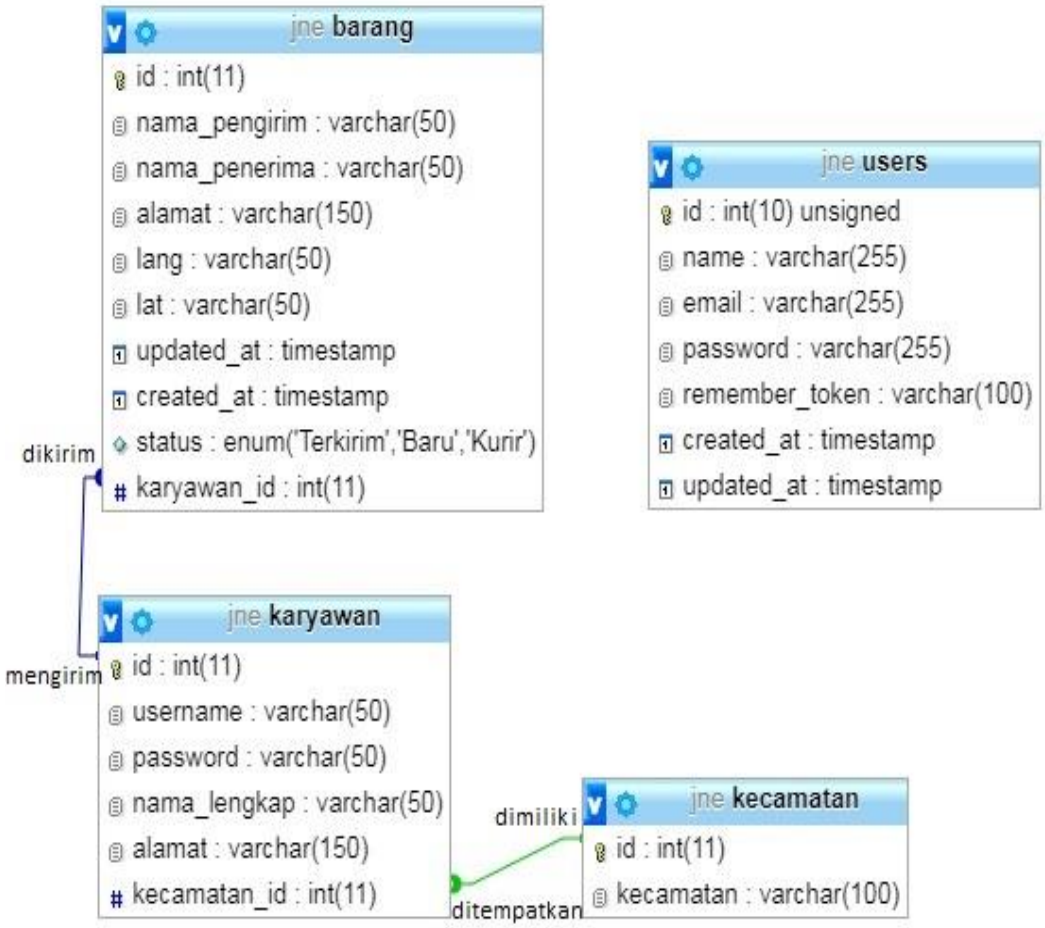
Class diagram menggambarkan hubungan antara kelas yang digunakan untuk membangun suatu sistem. Dalam paradigma OOP (*Object Oriented Programming*) terdapat 3 jenis kelas yaitu *model*, *view* dan *controller*. Berdasarkan *sequence* diagram yang telah dibangun, *class* diagram Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford terdiri atas 4 *class* model, 10 *class view* dan 4 *class controller*.



Gambar 4. 25 Class Diagram

4.4.7 *Entity Relationship Diagram*

(ERD) Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford merupakan gambaran komponen dan struktur *database* yang saling berhubungan untuk digunakan dalam pembuatan sistem. ERD yang diimplementasikan pada sistem ini terdiri dari empat entitas.



Gambar 4. 26 Entity Relationship Diagram

4.5 Penulisan Kode Program

Desain yang telah dibuat akan diimplemantasikan ke dalam kode program. Beberapa hal yang dilakukan dalam tahap implementasi antara lain:

a. Penulisan kode program (*coding*) menggunakan bahasa pemograman *Page Hyper Text Pre-Processor (PHP), HyperText Markup Language (HTML), Cascading Style Sheet (CSS), J-Query dan* dengan bantuan *framework Laravel*.

b. Manajemen basis data menggunakan *DBMS MySQL.*

Kode Program metode Optimasi Rute Pengiriman Barang Menggunakan Algoritma Bellman Ford sebagai berikut :

1. Public function test($date,$karyawan_id) digunakan untuk mengambil data sesuai dengan id karyawan dan tanggal. Kode program pada nomor 1-15 adalah function dari cara penentuan dan pembagian barang yang akan dikirim.
2. Public function php_permutasi($items, $perms = array( ),$d) digunakan untuk melakukan perhitungan rute terpendek pengiriman barang JNE. Kode program pada nomor 16-33 adalah cara perhitungan untuk menentukan nilai rute terpendek barang yang akan dikirim.

```
1. public function test($date,$karyawan_id){
2.          $d   =   Barang::where(DB::raw('DATE(created_at)'),'=',$date)-
    >where('karyawan_id','=',$karyawan_id)->orwhere('id','=',5)->get();
3.          // dd($d->toArray());
4.          $arr = json_decode($d);
5.          // dd($arr);
6.          $items = array();
7.          for ($i=0; $i < count($arr); $i++) {
                 $items[] = $i;
             }
8.          // dd($items);
9.          $data = $this->php_permutasi($items,array(),$d);
10.         // echo $this->ank;
```

```
        // dd($this->dat);
        // dd($data);
        $mine = $this->dat;
        // dd($mine);
        return view('hasil',compact('mine'));
    }

    public function php_permutasi($items, $perms = array( ),$d) {
         if (empty($items)) {
            // print $perms[0];
            // echo $d[0]->lang;
            $total = 0.0;
            $arr = array();
            for ($i=0; $i < count($perms)-1 ; $i++) {
                 $subtotal=        $this->getDistance($d[$perms[$i]]->lat,$d[$perms[$i]]->lang,$d[$perms[$i+1]]->lat,$d[$perms[$i+1]]->lang);
                $total += $subtotal;
                $obj = array();
                $obj[] = $d[$perms[$i]]->nama_penerima;
                $obj[] = $d[$perms[$i]]->id;
                $obj[] = $d[$perms[$i+1]]->nama_penerima;
                $obj[] = $d[$perms[$i+1]]->id;
                $obj[] = $subtotal;
                $obj[] = $d[$perms[$i]]->lat;
                $obj[] = $d[$perms[$i]]->lang;
                $obj[] = $d[$perms[$i+1]]->lat;
                $obj[] = $d[$perms[$i+1]]->lang;
                $arr[] = $obj;
        }
```

c. Penulisan kode program fitur dari Sistem Informasi Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford.

1. *View Login*

Penulisan kode *View Login* terletak di class *view login. View Login* adalah sebuah fitur yang menampilkan form login user, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 1.

2. *View Register*

Penulisan kode *View Register* terletak di *class view register*. *View Register* adalah sebuah fitur yang menampilkan halaman *register user*, dalam hal ini yang memiliki hak akses adalah admin. Penulisan kode tersebut dapat dilihat pada lampiran 2.

3. *View* Barang

Penulisan kode *View* Barang terletak di *class view* barang. *View* Barang adalah sebuah fitur yang menampilkan halaman barang, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 3.

4. *View Create* Barang

Penulisan kode *View Create* Barang terletak di class *view create* barang. *View Create* Barang adalah sebuah fitur yang menampilkan halaman barang, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 4.

5. *View Create* Karyawan

Penulisan kode *View Create* Karyawan terletak di class *view create* karyawan. *View Create* Karyawan adalah sebuah fitur yang menampilkan halaman pembuatan ID karyawan, dalam hal ini yang memiliki hak akses adalah admin .Penulisan kode tersebut dapat dilihat pada lampiran 5.

6. *View Detail* Pembagian

Penulisan kode *View Detail* Pembagian terletak di class *view detail* pembagian. *View detail* Pembagian adalah sebuah fitur yang menampilkan halaman pembagian, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 6.

7. *View Home*

Penulisan kode *View Home* terletak di class *view home. View home* adalah sebuah fitur yang menampilkan halaman utama, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 7.

8. *View* Kecamatan

Penulisan kode *View* Kecamatan terletak di class *view* pada *function view kecamatan. View kecamatan* adalah sebuah fitur yang menampilkan halaman untuk melihat kecamatan, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 8.

9. Karyawan *Controller*

Penulisan kode Karyawan *Controller* terletak di class *controller* pada *function* karyawan *controller*. Kecamatan *Controller* adalah sebuah fitur yang menampilkan halaman untuk melihat tugas dari karyawan, dalam hal ini yang memiliki hak akses adalah admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran 9.

10. Pembagian *Controller*

Penulisan kode Pembagian *Controller* terletak di class *controller* pada *function* pembagian *controller*. Pembagian *Controller* adalah sebuah fitur yang menampilkan halaman saat admin membagikan tugas pada karyawan, dalam hal ini yang memiliki hak akses adalah admin. Penulisan kode tersebut dapat dilihat pada lampiran `10.

11. *Model*

Penulisan kode *model* terletak pada *function* pembagian *model*. *Model* adalah sebuah fungsi untuk mengambil, memasukkan dan mengupdate database. Penulisan kode tersebut dapat dilihat pada lampiran `11.

12. Karyawan *Model*

Penulisan kode Karyawan Model terletak di class karyawan pada *function* model. Karyawan model adalah fungsi untuk mengambil, memasukkan dan mengupdate database karyawan. Penulisan kode tersebut dapat dilihat pada lampiran `12.

13. Kecamatan *Model*

Penulisan kode Kecamatan *Model* terletak di *class* kecamatan pada *function* model. Karyawan model adalah fungsi untuk mengambil, memasukkan dan mengupdate database kecamatan. Penulisan kode tersebut dapat dilihat pada lampiran `13.

14. *User Model*

Penulisan kode *User Model* terletak di *class user* pada *function model*. *User model* adalah fungsi untuk mengambil, memasukkan dan mengupdate database admin dan karyawan. Penulisan kode tersebut dapat dilihat pada lampiran `14.

4.6 Pengujian

4.6.1 Metode *Black Box*

Black box testing merupakan metode pengujian perangkat lunak yang memeriksa fungsionalitas dari aplikasi yang berkaitan dengan struktur internal atau kerja. Metode ini memfokuskan pada keperluan fungsionalitas dari software. Pengujian *black box* pada Optimasi Rute Pengiriman Barang menggunakan Algoritma Bellman Ford dilakukan untuk mengetahui masukan dan keluaran dari sistem sesuai dengan kebutuhan fungsional atau tidak. Pengujian dilakukan pada setiap *usecase*. Hasil pengujian *black box* sebagai berikut:

A. Pengujian Black Box 1 pada Tanggal 7 Desember 2017

1. *Use Case* Scenario Masuk

Nama	Masuk
Aktor	Admin, Kurir
Pre-Kondisi	Aktor belum masuk, halaman masuk
Pra-Kondisi	Aktor sudah masuk, halaman utama
SKENARIO NORMAL “Masuk”	

Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Membuka halaman website Optimasi Rute Pengiriman Barang		✓	
	2. Menampilkan formulir masuk a. E-Mail Address, varchar (255) b. Password, varchar (255)	✓	
3. Mengisi formulir masuk		✓	
4. Menekan tombol hak akses (admin atau karyawan)		✓	
5. Menekan tombol Login		✓	
	6. Menampilkan halaman utama sesuai dengan hak akses	✓	
SCENARIO ALTERNATIF			

“Email dan password tidak diisi			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
7. Menekan tombol Login		✓	
	8. Menampilkan notifikasi “Please fill out this field”	✓	

SCENARIO ALTERNATIF “Email dan password salah”			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
9. Mengisi formulir masuk		✓	
10. Menekan tombol Login		✓	
	11. Menampilkan formulir masuk a. E-Mail Address varchar (255) b. Password varchar (255)	✓	

2. *Use Case Scenario* Memasukkan Data Barang

Nama	Memasukkan Data Barang			
Aktor	Admin			
Pre-Kondisi	Admin berhasil masuk ke halaman utama			
Pra-Kondisi	Admin berhasil memasukkan data barang, halaman barang			
	SKENARIO NORMAL "Memasukkan Data Barang"			
Aktor	Sistem	Testing		
			Sesuai	Tdak Sesuai
1. Menekan menu barang			✓	
2. Menekan tombol tambah barang			✓	
	3. Menampilkan formulir pengisian data barang : a. Nama Pengirim, varchar (50) b. Nama Penerima, varchar (50)	✓		

		c. Alamat, varchar (150) d. Kecamatan, varchar (100) e. Pilih Karyawan, varchar (50)		
4. Mengisi formulir data barang			✓	
5. Menekan tombol simpan			✓	
	6. Menampilkan halaman data barang		✓	
	SCENARIO ALTERNATIF "Tipe data salah"			
Aktor	Sistem	Testing		
			Sesuai	Tidak Sesuai
7. Mengisi formulir data barang			✓	

8. .Menekan tombol simpan		✓	
	9. Menampilkan formulir pengisian data barang : a. Nama Pengirim, varchar (50) b. Nama Penerima, varchar (50) c. Alamat, varchar (150) d. Kecamatan, varchar (100) e. Pilih Karyawan ,varchar (50)	✓	
SCENARIO ALTERNATIF “Salah satu kolom tidak diisi”			

Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
10. Mengisi formulir data barang		✓	
11. Menekan tombol simpan		✓	
	12. Menampilkan formulir pengisian data barang : a. Nama Pengirim, varchar (50) e) Nama Penerima, varchar (50) f) Alamat, varchar (150)	✓	

	g) Kecamatan, varchar (100) h) Pilih Karyawan, varchar (50)			

3. Use Case Scenario Menghapus Data Barang

Nama	Menghapus Data Barang		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk ke halaman utama		
Pra-Kondisi	Admin berhasil menghapus data barang, halaman barang		
SKENARIO NORMAL "Menghapus Data Barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu barang		✓	
	2. Menampilkan data barang: b. ID c. Nama Pengirim d. Nama Penerima e. Alamat	✓	
3. Menekan tombol hapus		✓	

	4. Menampilkan data barang: a. ID b. Nama Pengirim d. Nama Penerima e. Alamat	✓	

4. *Use Case Scenario* Melihat Rute Terpendek Pengiriman Barang

Nama	Melihat rute terpendek pengiriman barang
Aktor	Admin, Kurir
Pre-Kondisi	Aktor sudah berhasil masuk, halaman utama
Pra-Kondisi	Aktor sudah berhasil melihat rute terpendek pengiriman barang, halaman rute terpendek

SKENARIO NORMAL

"Melihat Rute Terpendek Pengiriman Barang"

Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu pembagian			✓
	2. Menampilkan data pembagian barang: a.No b.Tanggal		✓
3. Menekan tombol detail			✓

pada sebelah kanan tanggal pengiriman yang ingin dicari		
	4. Menampilkan tabel data kurir: a. No b. Nama kurir c. Kecamatan	✓
5. Menekan tombol pekerjaan pada sebelah kanan nama kurir		✓
	6. Menampilkan tabel daftar data barang disertai dengan akumulasi jarak: a. No b. Kurir c. Menuju d. Jarak	✓
7. Menekan nama paling atas pada kolom menuju		✓
SCENARIO ALTERNATIF “TIdak memiliki Antrian Pekerjaan”		

Aktor	Sistem	Testing	
		Sesuai	Tdak Sesuai
8. Menekan tombol pekerjaan pada sebelah kanan nama kurir			✓
	9. Menampilkan halaman eror		✓
SCENARIO ALTERNATIF “TIdak Memiliki Detail Pekerjaan”			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
10. Menekan tombol detail pada sebelah kanan tanggal pengiriman yang ingin dicari			✓
	8. Menampilkan halaman eror		✓

5. *Use Case Scenario* Memasukkan Data Kurir

Nama	Memasukkan Data Kurir
Aktor	Admin
Pre-Kondisi	Aktor sudah berhasil masuk, halaman utama
Pra-Kondisi	Aktor sudah berhasil memasukkan data kurir, halaman kurir

SKENARIO NORMAL "Memasukkan Data Barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu kurir		✓	
2. Menekan tombol tambah kurir		✓	
	3. Menampilkan formulir pengisian data kurir : a. Username, varchar (50) b.Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) f. Penempatan Kecamatan, int (11)	✓	
4. Mengisi formulir data kurir		✓	
5. Menekan tombol simpan			
	6. Menampilkan halaman data kurir	✓	
SCENARIO ALTERNATIF "Tipe data salah"			
Aktor	Sistem	Testing	

		Sesuai	Tidak Sesuai
7. Mengisi formulir data kurir		✓	
8. Menekan tombol simpan		✓	
	9. Menampilkan formulir pengisian data kurir : a. Username,varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)	✓	
SCENARIO ALTERNATIF "Salah satu kolom tidak diisi"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
10. Mengisi formulir data kurir		✓	
11. Menekan tombol simpan		✓	
	12. Menampilkan formulir pengisian data kurir:	✓	

	a. Username, varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)	

6. *Use Case Scenario* Menghapus Data Kurir

Nama	Menghapus Data Kurir		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk ke halaman utama		
Pra-Kondisi	Admin berhasil menghapus data kurir halaman kurir		
SKENARIO NORMAL "Menghapus Data Barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu kurir		✓	
	2. Menampilkan data kurir: a. ID b. Username c. Nama Lengkap d. Alamat	✓	

	e. Kecamatan	
3. Menekan tombol hapus		✓
	4. Menampilkan data kurir: a. ID b. Username c. Nama Lengkap d. Alamat e. Kecamatan	✓

7. *Use Case Scenario* Melihat data kurir

Nama	Melihat data kurir
Aktor	Admin
Pre-Kondisi	Admin berhasil masuk, halaman utama
Pra-Kondisi	Admin berhasil melihat data kurir, halaman kurir

SKENARIO NORMAL "Melihat data kurir"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu kurir		✓	
	2. Menampilkan data kurir yang berisi : a. ID b. Username c. Nama Lengkap d. Alamat	✓	

8. *Use Case Scenario* Melihat Daftar Pengiriman Barang

Nama	Melihat daftar pengiriman barang		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk, halaman utama		
Pra-Kondisi	Admin berhasil melihat daftar pengiriman barang, halaman barang		
SKENARIO NORMAL "Melihat daftar pengiriman barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu barang		✓	
	2. Menampilkan tabel pengiriman barang yang berisi : a. ID b. Nama Pengirim c. Nama Penerima d. Alamat	✓	

9. *Use Case Scenario* Melihat Daftar Pembagian Barang

Nama	Melihat daftar pembagian barang		
Aktor	Kurir		
Pre-Kondisi	Kurir berhasil masuk, halaman utama		
Pra-Kondisi	Kurir berhasil melihat daftar pembagian barang, halaman hasil pembagian barang		
SKENARIO NORMAL "Melihat daftar pembagian barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan tombol detail			✓
	2. Menampilkan tabel hasil pembahian barang yang berisi: a. No b. Kurir c. Menuju d. Jarak		✓

10. *Use Case Scenario* Mencari Data Tanggal Pengiriman Barang

Nama	Mencari data tanggal pengiriman barang		
Aktor	Admin, Kurir		
Pre-Kondisi	Admin berhasil masuk ke halaman utama		
Pra-Kondisi	Admin berhasil melakukan pencarian, halaman pembagian		
SKENARIO NORMAL "Mencari data tanggal pengiriman barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai

1. Menekan menu pembagian		✓	
	2. Menampilkan tabel pembagian barang: a. No b. Tanggal	✓	
3. Mengisi kolom search			
	4. Menampilkan hasil pencarian sesuai dengan tanggal yang diisikan pada kolom search: a. No b. Tanggal	✓	

11. *Use Case Scenario* Keluar

Nama	Keluar		
Aktor	Admin, Kurir		
Pre-Kondisi	Aktor berhasil masuk, halaman utama		
Pra-Kondisi	Aktor berhasil keluar, halaman masuk		
SKENARIO NORMAL "Keluar"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan tombol logout		✓	

	2. Menampilkan formulir masuk yang berisi: a. E-mail Addres, varchar (255) b. Password, varchar (255)	✓	

B. Pengujian *Black Box* 2 pada Tanggal 27 April 2018

1. *Use Case* Scenario Masuk

Nama	Masuk		
Aktor	Admin, Kurir		
Pre-Kondisi	Aktor belum masuk, halaman masuk		
Pra-Kondisi	Aktor sudah masuk, halaman utama		
SKENARIO NORMAL "Masuk"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Membuka halaman website Optimasi Rute Pengiriman Barang		✓	
	2. Menampilkan formulir masuk a. E-Mail Address, varchar (255) b. Password, varchar (255)	✓	
3. Mengisi formulir masuk		✓	
4. Menekan tombol hak		✓	

akses (admin atau karyawan)			
5. Menekan tombol Login		✓	
	6. Menampilkan halaman utama sesuai dengan hak akses	✓	
SCENARIO ALTERNATIF “Email dan password tidak diisi			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
7. Menekan tombol Login		✓	
	8. Menampilkan notifikasi “Please fill out this field”	✓	
SCENARIO ALTERNATIF “Email dan password salah”			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
9. Mengisi formulir masuk		✓	

10. Menekan tombol Login		✓
	11. Menampilkan formulir masuk a. E-Mail Address varchar (255) b. Password varchar (255)	✓

2. *Use Case Scenario* Memasukkan Data Barang

Nama	Memasukkan Data Barang		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk ke halaman utama		
Pra-Kondisi	Admin berhasil memasukkan data barang, halaman barang		
SKENARIO NORMAL "Memasukkan Data Barang"			
Aktor	Sistem	Testing	
		Sesuai	Tdak Sesuai
1.Menekan menu barang		✓	
2. Menekan tombol tambah barang		✓	
	3. Menampilkan formulir pengisian data barang : a. NamaPengirim, varchar (50) b. Nama Penerima, varchar (50) c. Alamat, varchar (150) d. Kecamatan, varchar (100) e. Pilih Karyawan, varchar (50)	✓	
4.Mengisi		✓	

Formulir Data barang			
5. Menekan tombol simpan		✓	
	6. Menampilkan halaman data barang	✓	
SCENARIO ALTERNATIF "Tipe data salah"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
7. Mengisi formulir data barang		✓	
8. Menekan tombol simpan		✓	
	9. Menampilkan formulir pengisian data barang : a. Nama Pengirim, varchar (50) d. Nama Penerima, varchar (50) e. Alamat, varchar (150)	✓	

	f. Kecamatan, varchar (100) g. Pilih Karyawan, varchar (50)		
SCENARIO ALTERNATIF "Salah satu kolom tidak diisi"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Mengisi formulir data barang		✓	
2. Menekan tombol simpan		✓	
	5. Menampilkan formulir pengisian data barang : a. Nama Pengirim, varchar (50) b. Nama Penerima, varchar (50) c. Alamat, varchar (150) d. Kecamatan, varchar (100) e. Pilih Karyawan, varchar (50)	✓	

3. *Use Case Scenario* Menghapus Data Barang

Nama	Menghapus Data Barang		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk ke halaman utama		
Pra-Kondisi	Admin berhasil menghapus data barang, halaman barang		
SKENARIO NORMAL "Menghapus Data Barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu barang		✓	
	2. Menampilkan data barang: a. ID b. Nama Pengirim c. Nama Penerima d. Alamat	✓	
3. Menekan tombol hapus		✓	
	4. Menampilkan data barang: a. ID b. Nama Pengirim c. Nama Penerima d. Alamat	✓	

4. .*Use Case Scenario* Melihat Rute Terpendek Pengiriman Barang

Nama	Melihat rute terpendek pengiriman barang		
Aktor	Admin, Kurir		
Pre-Kondisi	Aktor sudah berhasil masuk, halaman utama		
Pra-Kondisi	Aktor sudah berhasil melihat rute terpendek pengiriman barang, halaman rute terpendek		
SKENARIO NORMAL "Melihat Rute Terpendek Pengiriman Barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu pembagian		✓	
	2. Menampilkan data pembagian barang: a. No b. Tanggal	✓	
3. Menekan tombol detail pada sebelah kanan tanggal pengiriman yang ingin dicari		✓	
	4. Menampilkan tabel data kurir a. No		✓

	b. Nama kurir c. Kecamatan		
5. Menekan tombol pekerjaan pada sebelah kanan nama kurir			✓
	6. Menampilkan tabel daftar data barang disertai dengan akumulasi jarak a. No b. Kurir c. Menuju d. Jarak		✓
7. Menekan nama paling atas pada kolom menuju			✓
SCENARIO ALTERNATIF "TIdak memiliki Antrian Pekerjaan"			
Aktor	Sistem	Testing	
		Sesuai	Tdak Sesuai
8. Menekan tombol pekerjaan pada sebelah kanan nama kurir		✓	

	9. Menampilkan halaman eror	✓	
SCENARIO ALTERNATIF “TIdak Memiliki Detail Pekerjaan”			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
10. Menekan tombol detail pada sebelah kanan tanggal pengiriman yang ingin dicari		✓	
	11. Menampilkan halaman eror	✓	

5. *Use Case Scenario* Memasukkan Data Kurir

Nama	Memasukkan Data Kurir		
Aktor	Admin		
Pre-Kondisi	Aktor sudah berhasil masuk, halaman utama		
Pra-Kondisi	Aktor sudah berhasil memasukkan data kurir, halaman kurir		
SKENARIO NORMAL “Memasukkan Data Barang”			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai

1. Menekan menu kurir		✓	
2. Menekan tombol tambah kurir		✓	
	3. Menampilkan formulir pengisian data kurir : a. Username, varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)	✓	
4. Mengisi formulir data kurir		✓	
5. Menekan tombol simpan			
	6. Menampilkan halaman data kurir	✓	
SCENARIO ALTERNATIF "Tipe data salah"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
7. Mengisi formulir data kurir		✓	

8. Menekan tombol simpan		✓	
	9. Menampilkan formulir pengisian data kurir: a. Username, varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)	✓	

SCENARIO ALTERNATIF "Salah satu kolom tidak diisi"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
10. Mengisi formulir data kurir		✓	
11. Menekan tombol simpan		✓	

	12. Menampilkan formulir pengisian data kurir : a. Username, varchar (50) b. Password, varchar (50) c. Nama lengkap, varchar (50) d. Alamat, varchar (150) e. Penempatan Kecamatan, int (11)	✓

6. *Use Case Scenario* Menghapus Data Kurir

Nama	Menghapus Data Kurir		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk ke halaman utama		
Pra-Kondisi	Admin berhasil menghapus data kurir halaman kurir		
	SKENARIO NORMAL "Menghapus Data Barang"		
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu kurir		✓	
	2. Menampilkan data kurir: a. ID	✓	

		b. Username c. Nama Lengkap d. Alamat e. Kecamatan	
3.	Menekan tombol hapus		✓
		4. Menampilkan data kurir: a. ID b. Username c. Nama Lengkap d. Alamat e. Kecamatan	✓

7. *Use Case Scenario* Melihat Data Kurir

Nama	Melihat data kurir		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk, halaman utama		
Pra-Kondisi	Admin berhasil melihat data kurir, halaman kurir		
SKENARIO NORMAL "Melihat data kurir"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu kurir		✓	
	2. Menampilkan data kurir yang berisi : a. ID b. Username c. Nama Lengkap d. Alamat	✓	

8. *Use Case Scenario* Melihat Daftar Pengiriman Barang

Nama	Melihat daftar pengiriman barang		
Aktor	Admin		
Pre-Kondisi	Admin berhasil masuk, halaman utama		
Pra-Kondisi	Admin berhasil melihat daftar pengiriman barang, halaman barang		
SKENARIO NORMAL "Melihat daftar pengiriman barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai

1. Menekan menu barang			✓
	2. Menampilkan tabel pengiriman barang yang berisi : a. ID b. Nama Pengirim c. Nama Penerima d. Alamat		✓

9. *Use Case Scenario* Melihat Daftar Pembagian Barang

Nama	Melihat daftar pembagian barang		
Aktor	Kurir		
Pre-Kondisi	Kurir berhasil masuk, halaman utama		
Pra-Kondisi	Kurir berhasil melihat daftar pembagian barang, halaman hasil pembagian barang		
SKENARIO NORMAL "Melihat daftar pembagian barang"			
Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan tombol detail		✓	
	2. Menampilkan tabel hasil pembahian barang yang berisi: a. No b. Kurir c. Menuju d. Jarak	✓	

10. *Use Case Scenario* Mencari Data Tanggal Pengiriman Barang

Nama	Mencari data tanggal pengiriman barang
Aktor	Admin, Kurir
Pre-Kondisi	Admin berhasil masuk ke halaman utama
Pra-Kondisi	Admin berhasil melakukan pencarian, halaman pembagian
SKENARIO NORMAL "Mencari data tanggal pengiriman barang"	

Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan menu pembagian		✓	
	2. Menampilkan tabel pembagian barang : a. No b. Tanggal	✓	
3. Mengisi kolom *search*			
	4. Menampilkan hasil pencarian sesuai dengan tanggal yang diisikan pada kolom search: a. No b. Tanggal	✓	

11. *Use Case Scenario* Keluar

Nama	Keluar
Aktor	Admin, Kurir
Pre-Kondisi	Aktor berhasil masuk, halaman utama
Pra-Kondisi	Aktor berhasil keluar, halaman masuk
SKENARIO NORMAL "Keluar"	

Aktor	Sistem	Testing	
		Sesuai	Tidak Sesuai
1. Menekan tombol logout		✓	
	2. Menampilkan formulir masuk yang berisi: a. E-mail Addres, varchar (255) b. Password, varchar (255)	✓	

4.6.2 White Box

Pengujian sistem dengan metode *white box* dilakukan untuk menguji sistem dari segi desain dan kode program. Pengujian bertujuan untuk mengevaluasi apakah sistem mampu menghasilkan fungsi, inputan, dan keluaran yang sesuai dengan spesifikasi dari kebutuhan sistem. Tahapan pengujian metode *white box* meliputi:

1. *Listing* program

Listing program merupakan kumpulan baris kode yang diuji.Setiap langkah dari kode-kode yang ada diberi nomor baik menjalankan statement biasa atau penggunaan kondisi dalam program.

2. Diagram alir

Diagram alir merupakan notasi yang digunakan untuk merepresentasikan aliran kontrol yang digambarkan dari hasil penomoran dari *listing* program. Diagram alir digambarkan dengan *node* (simpul) yang dihubungkan dengan *edge* (garis).

3. Kompleksitas siklomatik (*cyclomatic complexity*)

Kompleksitas siklomatik merupakan metrik perangkat lunak yang menyediakan ukuran kuantitatif dari kompleksitas logis suatu program. Kompleksitas

siklomatik mendefinisikan jumlah jalur independen dalam suatu program. Perhitungan kompleksitas siklomatik menggunakan persamaan 4

$$V(G) = E - N + 2 \quad (4)$$

Keterangan:

V(G) = Kompleksitas siklomatik

E = Jumlah *edge* (garis)

N = Jumlah *node* (simpul)

4. Jalur independen (*Independent Path*)

Jalur independen adalah setiap jalur yang melalui program, menunjukkan setidaknya satu kumpulan pernyataan-pernyataan pemrosesan atau kondisi baru. Jalur independen dalam grafik alir bergerak setidaknya satu *edge* yang belum dilintasi sebelum jalur tersebut didefinisi.

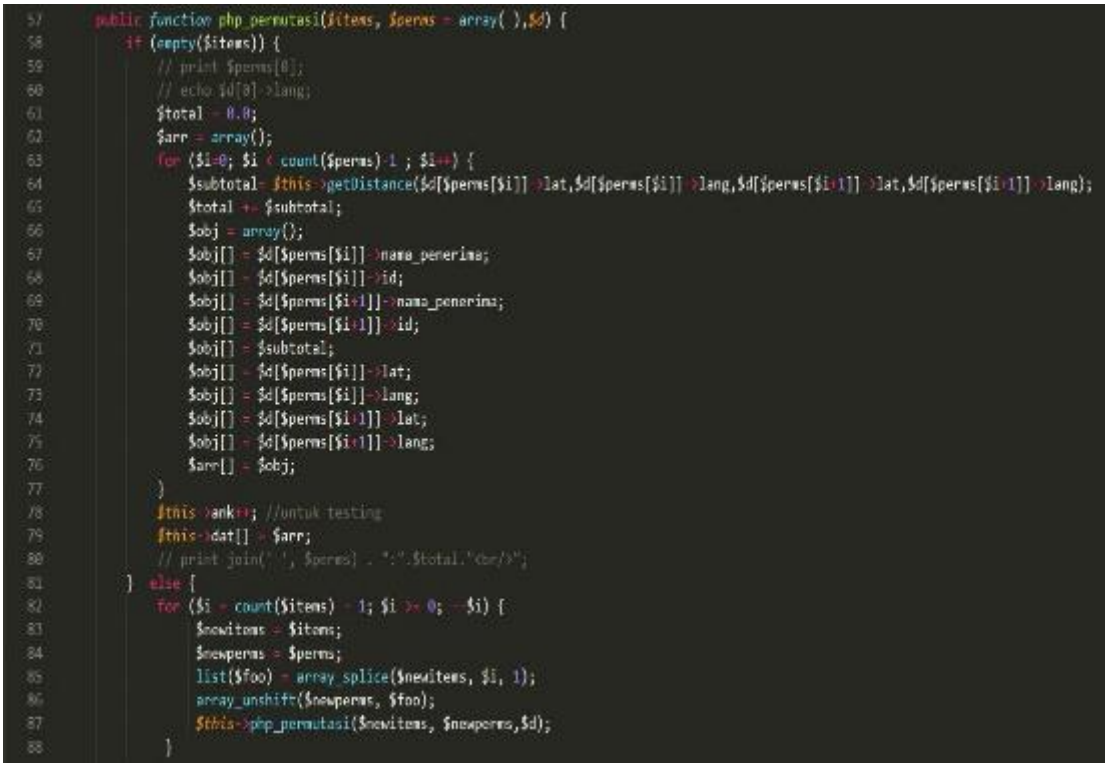
5. Pengujian Basis Tes (*Test Case*)

Pada bagian ini diberikan contoh data yang menggambarkan pelaksanaan jalur di basis tes. Data yang dieksekusi dimasukkan ke dalam grafik alir apakah sudah melewati basis tes yang tersedia. Sistem telah memenuhi syarat kelayakan perangkat lunak jika salah satu jalur yang dieksekusi setidaknya satu kali.

Pengujian optimasi rute pengiriman barang menggunakan algoritma Bellman Ford pada PembagianController dinilai dapat mewakili alur sistem informasi ini dijelaskan sebagai berikut:

Listing Program Pembagian *Controller* dapat dilihat pada Gambar 4.6 sampai Gambar 4.7 berikut :

6. Gambar *Listing Program*



```
public function php_permutasi($items, $perms = array( ),$d) {
    if (empty($items)) {
        // print $perms[0];
        // echo $d[0]->lang;
        $total = 0.0;
        $arr = array();
        for ($i=0; $i < count($perms)-1 ; $i++) {
            $subtotal= $this->getDistance($d[$perms[$i]]->lat,$d[$perms[$i]]->lang,$d[$perms[$i+1]]->lat,$d[$perms[$i+1]]->lang);
            $total += $subtotal;
            $obj = array();
            $obj[] = $d[$perms[$i]]->nama_penerima;
            $obj[] = $d[$perms[$i]]->id;
            $obj[] = $d[$perms[$i+1]]->nama_penerima;
            $obj[] = $d[$perms[$i+1]]->id;
            $obj[] = $subtotal;
            $obj[] = $d[$perms[$i]]->lat;
            $obj[] = $d[$perms[$i]]->lang;
            $obj[] = $d[$perms[$i+1]]->lat;
            $obj[] = $d[$perms[$i+1]]->lang;
            $arr[] = $obj;
        }
        $this->ank++; //untuk testing
        $this->dat[] = $arr;
        // print join(' ', $perms) . ":".$total."<br/>";
    } else {
        for ($i = count($items) - 1; $i >= 0; --$i) {
            $newitems = $items;
            $newperms = $perms;
            list($foo) = array_splice($newitems, $i, 1);
            array_unshift($newperms, $foo);
            $this->php_permutasi($newitems, $newperms,$d);
        }
```

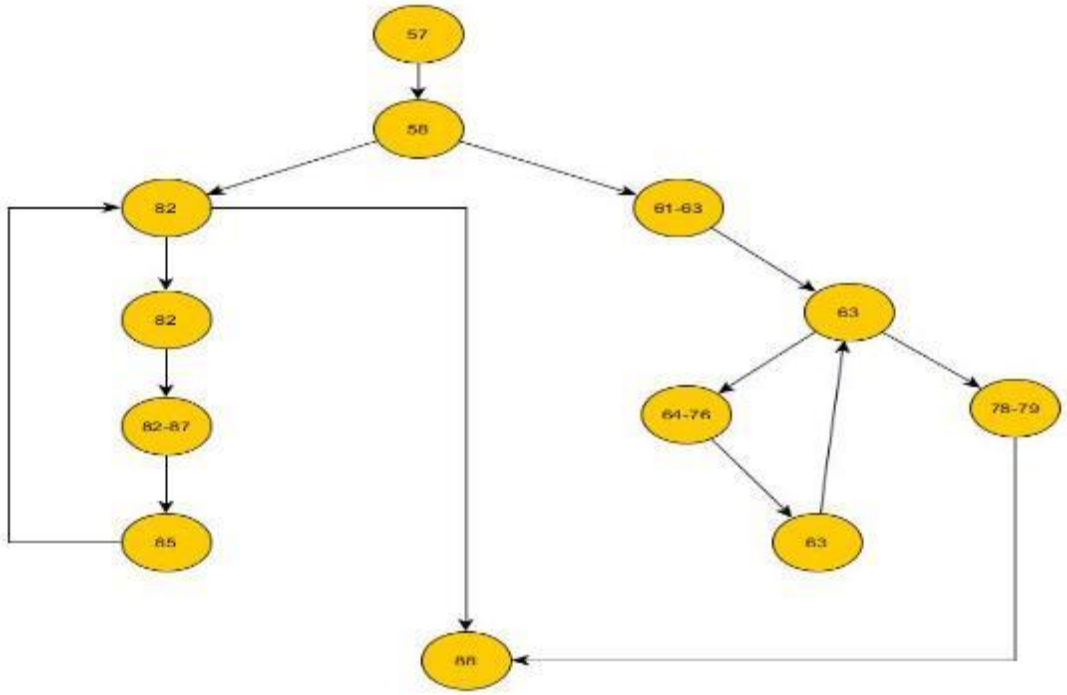
Gambar 4.26 *Listing* Program Permutasi

```
function getDistance($point1_lat, $point1_long, $point2_lat, $point2_long, $decimals = 2) {
    $degrees = rad2deg(acos((sin(deg2rad($point1_lat))*sin(deg2rad($point2_lat))) + (cos(deg2rad($point1_lat))*cos(deg2rad($point2_lat))*cos(deg2rad
        ($point1_long-$point2_long)))));
    $distance = $degrees*111133.84;
    return round($distance, $decimals);
}
```

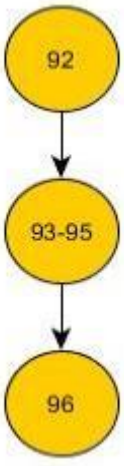
Gambar 4. 27 *Listing Program Get Distance*

7. Gambar Diagram Permutasi

Diagram Permutasi *class* model dapat dilihat pada Gambar 4.28 sampai Gambar 4.29 berikut:



Gambar 4. 29 Diagram Permutasi



Gambar 4. 30 Diagram *Distance*

8. Penjelasan Kompleksitas Siklomatik (*Cyclomatic Complexity*)

1. Fungsi program permutasi

V(G) = E – N + 2

V(G) = 12 – 14 + 2 = 10

2. Fungsi program *distance*

V(G) = E – N + 2

V(G) = 2 – 3 + 2 = 1

9. Penjelasan Pengujian Basis Test (*Test Case*)

Tabel 4. 31 *Test Case function* php_permutasi

function *php_permutasi*	
Jalur 1 : 57–58–61–63–64–88	
Kasus	Tidak bisa menghitung jarak
Target yang diharapkan	Dapat menghitung seluruh kemungkinan jarak yang akan dilewati
Hasil Pengujian	Berhasil
Jalur 1 : 57–58–61–63–64–76–78–79	
Kasus	Jarak yang ditempuh jauh
Target yang diharapkan	Jarak yang ditempuh menjadi lebih dekat karena dapat menghitung semua kemungkinan jarak

Tabel 4. 32*TestCase function getDistance*

function getDistance	
Jalur 1 : 92 – 93 – 95 – 96	
Kasus	Tidak mengetahui jarak antar titik

Target yang diharapkan	Mengetahui jarak antar titik yang akan yang akan dilewati
Hasil Pengujian	Berhasil

BAB 6. PENUTUP

Bab ini berisi kesimpulan dan saran dari peneliti tentang penelitian yang telah dilakukan. Kesimpulan dan saran tersebut diharapkan dapat digunakan sebagai acuan pada penelitian selanjutnya.

6.1 Kesimpulan

Berdasarkan analisis yang telah dilakukan oleh peneliti, dapat diambil kesimpulan sebagai berikut :

1. Algoritma Bellman Ford diimplementasikan untuk mencari rute terpendek pengiriman barang menggunakan Laravel. Selanjutnya dibutuhkan koordinat atau alamat barang penerima yang akan dikirim untuk dijadikan sebagai titik acuan dalam menghitung dan mengetahui rute terpendek dari semua alamat barang yang sudah ditentukan. Lokasi awal pengiriman ke titik pertama menempuh jarak 908.17 m, dilanjutkan dari titik pertama ke titik kedua menempuh jarak 243.48 m, selanjutnya dari titik dua ke titik 3 menempuh jarak 846.41 m. Hasil yang diperoleh dari perhitungan algoritma Bellman Ford yang merupakan rute terpendek adalah adalah 1998.06 m.
2. Perancangan aplikasi optimasi rute pengiriman barang ini berdasarkan algoritma Bellman Ford. Algoritma Bellman Ford digunakan dalam menghitung dan mencari rute terpendek pengiriman barang. Data barang yang akan dikirim diperoleh dari tempat penyimpanan barang atau gudang. Sebelum barang dikirim, barang tersebut didata terlebih dahulu. Admin menentukan barang yang akan dikirim oleh kurir berdasarkan lokasi atau daerahnya. Kurir memiliki daerah yang menjadi tugasnya dalam melakukan pengiriman barang. Setelah admin menentukan barang yang akan dikirim, kurir tersebut dapat mengikuti jalur mana saja yang harus dilewati untuk mengirim barang.

6.2 Saran

Adapun saran yang ditujukan untuk memberikan masukan yang lebih baik yaitu:

1. Di dalam tugas akhir ini, yang berwenang untuk mengatur pengiriman barang adalah admin. Kurir hanya melakukan tugasnya dalam mengirimkan barang kepada alamat penerima. Untuk penelitian selanjutnya bisa digunakan tambahan waktu dan jarak sebagai acuan pengiriman barang. Hal ini bisa membuat pengiriman barang lebih cepat dan efisien.
2. Untuk penelitan selanjutnya, kompleksitas dalam menentukan rute terpendek bisa ditambah, seperti waktu yang diperlukan, jarak yang ditempuh, ruas jalan yang dilewati, jumlah rambu lalu lintas, sehingga pengaplikasian sistem yang lebih kompleks algoritma Bellman Ford dapat diterapkan dengan baik.

DAFTAR PUSTAKA

Andana, G. 2009. Algoritma Bellman-Ford dalam Distance Vector Routing Protocol. Bandung

Retanto, Y. 2009. Algoritma Dijkstra dan Bellman-Ford dalam Pencarian Jalur Terpendek. Bandung

Handika U, S. 2010. Algoritma Bellman-Ford Sebagai Solusi Pencarian Akses Tercepat dalam Jaringan Komputer. Bandung

Kairurrazi B, B. 2009. Algoritma Dijkstra, Bellman-Ford, dan Floyd-Warshall Untuk Mencari Rute Terpendek dalam Suatu Graf. Bandung

Yohanes, A. 2014. Pencarian Rute Terpendek Menggunakan Algoritma Greedy. Semarang

Romelta, E. 2009. Metode Pencarian Lintasan Terpendek dalam Graf. Bandung

Ekaputra, A. 2014. Pencarian Rute Terpendek Menggunakan Algoritma Greedy. Semarang

Kristianto, B. 2012. Aplikasi Algoritma Bellman Ford dalam Meminimumkan Biaya Operasional Rute Penerbangan. Yogyakarta

LAMPIRAN

1. *View Login*

```
@extends('layouts.app2')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div    class="panel    panel-default"    style="margin-
top:300px;">
                <div class="panel-heading">Login</div>

                <div class="panel-body">
                    <form   class="form-horizontal"   method="POST"
action="{{ route('login') }}">
                        {{ csrf_field() }}

                        <div      class="form-group{{      $errors-
>has('email') ? ' has-error' : '' }}">
Lanjutan
                            Lanjutan
 <label    for="email"    class="col-md-4    control-label">E-Mail
Address</label>

                            <div class="col-md-6">
                                <input   id="email"   type="email"
class="form-control"   name="email"   value="{{   old('email')   }}"
required autofocus>

                                @if ($errors->has('email'))
                                    <span class="help-block">
                                        <strong>{{        $errors-
>first('email') }}</strong>
```

```
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div      class="form-group{{      $errors-
>has('password') ? ' has-error' : '' }}">
                            <label  for="password"  class="col-md-4
control-label">Password</label>

                            <div class="col-md-6">
                                <input               id="password"
type="password" class="form-control" name="password" required>

                                @if ($errors->has('password'))
                                    <span class="help-block">
                                        <strong>{{         $errors-
>first('password') }}</strong>

</span>
                                @endif
                            </div>
                        </div>

                        <div class="form-group">
                            <div class="col-md-6 col-md-offset-4">
                                <div class="checkbox">
                                    <label>
                                        <input      type="checkbox"
name="remember" {{ old('remember') ? 'checked' : '' }}> Remember Me
                                    </label>
                                </div>
                            </div>
```

```
                            </div>

                            <div class="form-group">
                                <div class="col-md-8 col-md-offset-4">
                                    <button  type="submit"  class="btn
        btn-primary">
                                        Login
                                    </button>

                                    <a  class="btn  btn-link"  href="{{
        route('password.request') }}">
                                        Forgot Your Password?
                                    </a>
          </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
@endsection
```

2. ***View Register***

```
@extends('layouts.app2')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-8 col-md-offset-2">
            <div class="panel panel-default">
                <div class="panel-heading">Register</div>
```

```
                <div class="panel-body">
                    <form   class="form-horizontal"   method="POST"
action="{{ route('register') }}">
                        {{ csrf_field() }}
      <div class="form-group{{ $errors->has('name') ? ' has-error'
: '' }}">
                            <label    for="name"    class="col-md-4
control-label">Name</label>

                            <div class="col-md-6">
                                <input    id="name"    type="text"
class="form-control" name="name" value="{{ old('name') }}" required
autofocus>

                                @if ($errors->has('name'))
                                    <span class="help-block">
                                        <strong>{{         $errors-
>first('name') }}</strong>
  </span>
                                @endif
                            </div>
                        </div>

                        <div      class="form-group{{      $errors-
>has('email') ? ' has-error' : '' }}">
                            <label   for="email"   class="col-md-4
control-label">E-Mail Address</label>

                            <div class="col-md-6">
                                <input   id="email"   type="email"
class="form-control"   name="email"   value="{{   old('email')   }}"
required>
```

```
                                @if ($errors->has('email'))
                                    <span class="help-block">
                                        <strong>{{        $errors-
>first('email') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>

                        <div      class="form-group{{      $errors-
>has('password') ? ' has-error' : '' }}">
                            <label  for="password"  class="col-md-4
control-label">Password</label>
Lanjutan
Lanjutan
 <div class="col-md-6">
                                <input               id="password"
type="password" class="form-control" name="password" required>

                                @if ($errors->has('password'))
                                    <span class="help-block">
                                        <strong>{{        $errors-
>first('password') }}</strong>
                                    </span>
                                @endif
                            </div>
                        </div>
                        <div class="form-group">
Lanjutan
Lanjutan
```

```
                                <label           for="password-confirm"
        class="col-md-4 control-label">Confirm Password</label>

                                <div class="col-md-6">
                                    <input       id="password-confirm"
        type="password"  class="form-control"  name="password_confirmation"
        required>
                                </div>
                            </div>

                            <div class="form-group">
                                <div class="col-md-6 col-md-offset-4">
                                    <button  type="submit"  class="btn
        btn-primary">
                                        Register
        </button>
                                </div>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>
@endsection
```

3. *View* Barang

```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-10">
            <div class="panel panel-primary">
                <div class="panel-heading">Dashboard
                <a   href="{{route('tambahBarang')}}"   class="btn   btn-
warning btn-sm pull-right btn-fill">Tambah Barang</a>
                </div>
                <div class="panel-body">
                    @if (session('status'))
                        <div class="alert alert-success">
                            {{ session('status') }}
                        </div>
                    @endif
                   Lanjutan
                   <table class="table table-responsive">
                        <thead>
                            <tr>
                                <th>ID</th>
                                <th>Nama Pengirim</th>
                                <th>Nama Penerima</th>
                                <th>Alamat</th>
                                <th>Action</th>
                            </tr>
                        </thead>
                        <tbody>
                            @foreach($data as $row)
                            <tr>
                                <td>{{$row->id}}</td>
```

```
                                <td>{{$row->nama_pengirim}}</td>
                                <td>{{$row->nama_penerima}}</td>
                                <td>{{$row->alamat}}</td>
    <td><a   href="{{url('Barang/Delete/'.$row->id)}}"   class="btn   btn-
danger btn-sm">Hapus</a></td>
                            </tr>
                            @endforeach
                        </tbody>
                    </table>
                </div>

            </div>
        </div>
    </div>
</div>
@endsection
```

4. *View Create* Barang

```
@extends('layouts.app')
@section('content')
<!--                    <script                    type="text/javascript"
src="http://maps.google.com/maps/api/js?sensor=false"></script> -->
<script                            async                            defer
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyBEguua9rQsZxWvqPRt
367bwlEmidbxHdI&callback=initialize&sensor=false"></script>
<script type="text/javascript">
    // var geocoder = new google.maps.Geocoder();
    function geocodePosition(pos) {
      document.getElementById('langitude').value = pos.lat();
      document.getElementById('latitude').value = pos.lng();
  //     geocoder.geocode({
  //       latLng: pos
```

```
  //   }, function(responses) {
  //       if (responses && responses.length > 0) {
  //         updateMarkerAddress(responses[0].formatted_address);
  //     } else {
  //              updateMarkerAddress('Cannot determine address at this
location.');
  //     }
  // });
}
function updateMarkerStatus(str) {
  document.getElementById('markerStatus').innerHTML = str;
}
function updateMarkerPosition(latLng) {
  document.getElementById('info').innerHTML = [
  latLng.lat(),
  latLng.lng()
  ].join(', ');
}
function updateMarkerAddress(str) {
  document.getElementById('address').innerHTML = str;
}
function initialize() {
  var latLng = new google.maps.LatLng(-8.1698703, 113.702621);
  var map = new google.maps.Map(document.getElementById('mapCanvas'), {
    zoom: 8,
    center: latLng,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  });
  var marker = new google.maps.Marker({
  Lanjutan
Lanjutan
  position: latLng,
```

```
    title: 'Point A',
    map: map,
draggable: true
  });
  // Update current position info.
  updateMarkerPosition(latLng);
  geocodePosition(latLng);
  // Add dragging event listeners.
  google.maps.event.addListener(marker, 'dragstart', function() {
    updateMarkerAddress('Dragging...');
  });
  google.maps.event.addListener(marker, 'drag', function() {
    updateMarkerStatus('Dragging...');
 updateMarkerPosition(marker.getPosition());
  });
  google.maps.event.addListener(marker, 'dragend', function() {
    updateMarkerStatus('Drag ended');
    geocodePosition(marker.getPosition());
  });
}
// Onload handler to fire off the app.
google.maps.event.addDomListener(window, 'load', initialize);
</script>
Lanjutan
Lanjutan
<style>
#mapCanvas {
  width: 100%;
  height: 400px;
  float: left;
}
#infoPanel {
  float: left;
```

```
 Lanjutan
 margin-left: 10px;
}
#infoPanel div {
  margin-bottom: 5px;
}
</style>
<div class="container">
  <div class="row">
    <div class="col-md-10">
      <div class="panel panel-success ">
        <div class="panel-heading">Input Barang
        </div>
        <div class="panel-body">
          @if (session('status'))
          <div class="alert alert-success">
            {{ session('status') }}
  </div>
Lanjutan
Lanjutan
          @endif
          <form action="{{route('createBarang')}}" method="post">
            {{csrf_field()}}
            <div class="form-group">
              <label class="col-md-3">Nama Pengirim</label>
              <div class="col-md-9">
                <input    type="text"    class="form-control    col-md-9"
name="nama_pengirim">
              </div>
            </div>
            <br>
            <br>
            <br>
```

```
            <div class="form-group">
              <label class="col-md-3">Nama Penerima</label>
              <div class="col-md-9">
                <input    type="text"    class="form-control    col-md-9"
name="nama_penerima">
                        Lanjutan
   </div>
            </div>
            <br>
            <br>
            <div class="form-group">
              <label class="col-md-3">Alamat</label>
<div class="col-md-9">
    <input type="text" class="form-control col-md-9" name="alamat">
              </div>
            </div>
            <br>
            <br>
            <div id="mapCanvas" class="form-control"></div>
            <br>
            <br>
            <div id="infoPanel" class="hidden">
              <b>Marker status:</b>
              <div     id="markerStatus"><i>Click     and     drag     the
marker.</i></div>
             Lanjutan
     <b>Current position:</b>
              <div id="info"></div>
              <b>Closest matching address:</b>
              <div id="address"></div>
            </div>
            <input type="hidden" id="langitude" name="lang">
            <input type="hidden" id="latitude" name="lat">
```

```
        <br>
            <hr>
            <div class="form-group">
              <label class="col-md-3">Kecamatan</label>
              <div class="col-md-9">
                <select  name="kecamatan_id"  id="kecamatan_id"  required
class="form-control col-md-9" onchange="getKaryawan();">
<option value="">Pilih Kecamatan</option>
                  @foreach($kec as $row)
                  <option                    value="{{$row->id}}">{{$row-
>kecamatan}}</option>
   @endforeach
                </select>
              </div>
            </div>
            <br>
            <br>
            <div class="form-group">
              <label class="col-md-3">Pilih Karyawan</label>
              <div class="col-md-9">
                <select  name="karyawan_id"  required  class="form-control
col-md-9" id="karyawan_id">
                  <option value="">Pilih Karyawan</option>
                </select>
              </div>
            </div>
            <div class="form-group">
  <div class="col-md-3"></div>
              <button type="submit" class="btn btn-md btn-success btn-fill
form-control">Simpan</button>
            </div>
          </form>
          <script>
```

```
            function getKaryawan(){
id = $('#kecamatan_id').val();
              url = '{{url("kar/")}}/'+id;
              $('#karyawan_id').empty();
              $('#karyawan_id').append('<option            value="">Pilih
Karyawan</option>');
              $.getJSON(url, function(data){
                data.forEach(function(element){
                  d                        =                        "<option
value='"+element.id+"'>"+element.nama_lengkap+"<option>";
                  // alert(d);
                  $('#karyawan_id').append(d);
   });
              });
            }
          </script>
        </div>
      </div>
    </div>
  </div>
</div>
@endsection
```

5. *View Create* Karyawan

```
@extends('layouts.app')
@section('content')
<div class="container">
       <div class="row">
              <div class="col-md-10">
                     <div class="panel panel-success ">
                            <div class="panel-heading">Input Barang
                            </div>
```

```
                        <div class="panel-body">
                              @if (session('status'))
                              <div class="alert alert-success">
                                    {{ session('status') }}
                              </div>
                              @endif
                              <form
action="{{route('createKaryawan')}}" method="post">
                                    {{csrf_field()}}
                                    <div class="form-group">
                                          <label       class="col-md-
3">Username</label>
                                          <div class="col-md-9">
                                                <input type="text"
class="form-control col-md-9" name="username">
                                          </div>
                                    </div>
                                    <br>
                                    <br>
                                    <br>
                                    <div class="form-group">
                                          <label       class="col-md-
3">Password</label>
                                          <div class="col-md-9">
                                                <input    type="text"
class="form-control col-md-9" name="password">
                                          </div>
                                    </div>
                                    <br>
                                    <br>
                                    <div class="form-group">
```

```
                                        <label       class="col-md-
3">Nama Lengkap</label>
                                        <div class="col-md-9">
                                              <input    type="text"
class="form-control col-md-9" name="nama_lengkap">
                                        </div>
                                  </div>
                                  <br>
                                  <br>
<div class="form-group">
                                        <label class="col-md-
3">Alamat</label>
                                        <div class="col-md-9">
                                              <input    type="text"
class="form-control col-md-9" name="alamat">
                                        </div>
                                  </div>
                                  <br>
                                  <br>
                                  <div class="form-group">
                                        <label       class="col-md-
3">Penempatan Kecamatan</label>
                                        <div class="col-md-9">
                                              <select
name="kecamatan_id" required class="form-control col-md-9">
                                                    <option
value="">Pilih Kecamatan</option>
                                                    @foreach($kec
as $row)
                                              <option
value="{{$row->id}}">{{$row->kecamatan}}</option>
                                                    @endforeach
                                              </select>
```

```
                                                    </div>
                                                </div>
<br>
                                                <br>

<div class="form-group">
                                                    <div         class="col-md-
3"></div>
                                                    <button       type="submit"
class="btn btn-md btn-success btn-fill form-control">Simpan</button>
                                                </div>
                                            </form>
                                        </div>
                                    </div>
                                </div>
                            </div>
</div>
@endsection
```

6. *View Detail* Pembagian

```
@extends('layouts.app')
@section('content')
<div class="container">
  <div class="row">
    <div class="col-md-12">
      <div class="panel panel-primary" id>
        <div class="panel-heading">Detail Pembagian Barang
          @if(Auth::user())
          <a href="{{route('tambahDetailPem',['id' => $id])}}" class="btn
btn-danger btn-sm pull-right btn-fill">Tambah</a>
          @endif
        </div>
```

```
        <div class="panel-body">
          @if (session('status'))
          <div class="alert alert-success">
            {{ session('status') }}
          </div>
          @endif

          <table id="table" class="table table-responsive">
            <thead>
              <tr>
                <th>ID Ekspedisi</th>
<th>ID Barang</th>
                <th>Nama Penerima</th>
                <th>Tanggal</th>
                @if(Auth::user())
                <th>Action</th>
                @endif
              </tr>
            </thead>
            <tbody>
              <?php $mine = $data->detail; ?>
              @foreach($mine as $row)
              <tr>
                <td>{{$row->id}}</td>
                <td>{{$row->barang_id}}</td>
                <td>{{$row->barang->nama_penerima}}</td>
                <td>{{$row->barang->created_at}}</td>
                @if(Auth::user())
                <td>
                  <a href="{{route('hapusDetailPembagian',['id' => $row-
>id])}}" class="btn btn-danger">Hapus</a>
                </td>
                @endif
```

```
            </tr>
            @endforeach
          </tbody>
        </table>
<br>
        <br>
        <a  href="{{route('Hasil',['id'  =>  $id])}}"  class="btn  btn-
warning btn-md pull-right btn-fill col-md-12">Hasil</a>
      </div>
    </div>
    <style>
    #map {
      height: 100%;
      width: 50%;
    }
  </style>
  <div id="map"></div>
  <div id="output">
  </div>
 </div>
</div>
 <!-- <div class="row">
   <div class="col-md-10">
     <div class="panel panel-primary">
       <div class="panel-heading">Jarak Antar Barang
       </div>
       <div class="panel-body">
         @if (session('status'))
<div class="alert alert-success">
           {{ session('status') }}
         </div>
         @endif
```

```
          <table id="table" class="table table-responsive">
            <thead>
              <tr>
                <th>No</th>
                <th>Nama Penerima 1</th>
                <th>Nama Penerima 2</th>
                <th>Jarak</th>
              </tr>
            </thead>
            <tbody id="data">
            </tbody>
          </table>
        </div>
      </div>
      <style>
      #map {
        height: 100%;
        width: 50%;
      }
    </style>
<div id="map"></div>
    <div id="output">

    </div>
  </div>
</div> -->
</div>
<!-- <script>
  var rad = function(x) {
    return x * Math.PI / 180;
  };

  var getDistance = function(p1lat,p1lang, p2lat,p2lang) {
```

```
    var R = 6378137; // Earth’s mean radius in meter
    var dLat = rad(p2lat - p1lat);
    var dLong = rad(p2lang - p1lang);
    var a = Math.sin(dLat / 2) * Math.sin(dLat / 2) +
      Math.cos(rad(p1lat)) * Math.cos(rad(p2lat)) *
      Math.sin(dLong / 2) * Math.sin(dLong / 2);
    var c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));
    var d = R * c;
    return d; // returns the distance in meter
  };
  $(document).ready(function(){
    var data = <?php echo json_encode($mine); ?>;
    var sample = [];
var hitung = [];
    k = 0;
    for(var i = 0;i<=data.length;i++){
      var a = 0;
      for(var j = i+1;j<=data.length-1;j++){
          a++;
          var sub = [[],[],[]];
          sub[0][0] = data[i]['barang']['id'];
          sub[0][1] = data[i]['barang']['nama_penerima'];
          sub[1][0] = data[j]['barang']['id'];
          sub[1][1] = data[j]['barang']['nama_penerima'];
          sub[2][0]                                                    =
getDistance(data[i]['barang']['lat'],data[i]['barang']['lang'],data[j]['b
arang']['lat'],data[j]['barang']['lang']);
          sample[k++] = sub;
          hitung[i] = a;
          // console.log(i+'-'+j);
          //
console.log(getDistance(data[i]['barang']['lat'],data[i]['barang']['lang'
],data[j]['barang']['lat'],data[j]['barang']['lang']));
```

```
      }
    }
        for(var i = 0;i<sample.length;i++){
        var                                ho                                =
"<tr><td>"+(i+1)+"</td><td>"+sample[i][0][1]+"</td><td>"+sample[i][1][1]+
"</t
d><td>"+Math.round(sample[i][2][0])+" m</td></tr>";
        $('#data').append(ho);
    }
    console.log(sample);
    console.log(hitung);
  });
  function abis(data){
  }
</script> -->
@endsection
```

7. *View Home*

```
@extends('layouts.app')
@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-10">
            <div class="panel panel-default">
                <div class="panel-heading">Dashboard</div>
      <div class="panel-body">

                    @if (session('status'))
                        <div class="alert alert-success">
                            {{ session('status') }}
                        </div>
```

```
                    @endif
                    You are logged in!
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

8. *View* Kecamatan

```
@extends('layouts.app')
@section('content')
<div class="container">
    <div class="row">
        <div class="col-md-12">

            <div class="panel panel-primary">
                <div class="panel-heading">Data Karyawan
                <a  href="{{route('tambahKaryawan')}}"  class="btn  btn-
warning btn-sm pull-right btn-fill">Tambah Karyawan</a>
                </div>
                <div class="panel-body">
                    @if (session('status'))
                        <div class="alert alert-success">
                            {{ session('status') }}
                        </div>
                    @endif
                    <table class="table table-responsive">
                        <thead>
                            <tr>
                                <th>No</th>
                                <th>Nama Karyawan</th>
```

```
                                <th>Kecamatan</th>
                                <th>Action</th>
  </tr>
                        </thead>
                        <tbody>
                            <?php $no = 1; ?>
                            @foreach($kecamatan as $row)
                            <tr>
                                <td>{{$no}}</td>
                                <td>{{$row->nama_lengkap}}</td>
                                <td>{{$row->kecamatan}}</td>
                                <td>
                                    <a  href="{{route('Hasil',['date'  =>
$date,'karyawan_id' => $row->id])}}" class="btn btn-primary">Pekerjaan</a>
                                </td>
                            </tr>
                            @endforeach
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </div>
</div>
@endsection
```

9. Karyawan Controller

```
<?php
namespace App\Http\Controllers;
use Illuminate\Http\Request;
use App\Karyawan;
use App\Kecamatan;
```

```
class KaryawanController extends Controller
{
    public function index()
 Lanjutan
Lanjutan
   {
        $data = Karyawan::all();
        return view('karyawan',compact('data'));
    }
    public function home(){
        return $this->masterPembagian(session('auth')->id);
    }
    public function login(Request $request){
        // print_r($request->all());
        $data     =     Karyawan::where('username','=',$request->email)-
>where('password','=',$request->password)->get();
        if (count($data) == 1) {
            session(['auth' => $data[0]]);
return redirect()->route('DashKaryawan');
        }else{
            return back()->withError('error',$request);
        }
        // echo count($data);
    }
    public function masterPembagian($id){
        return redirect()->route('Pembagian3',['id' => $id]);
    }
    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
```

```
    {
      Lanjutan
Lanjutan
  $kec = Kecamatan::all();
        return view('createKaryawan',compact('kec'));
    }
    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
* @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $data = new Karyawan;
        $data->username = $request->username;
        $data->password = $request->password;
        $data->nama_lengkap = $request->nama_lengkap;
        $data->alamat = $request->alamat;
        $data->kecamatan_id = $request->kecamatan_id;
        $data->save();
        return redirect()->route('HomeKaryawan');
    }
    public function kec($id){
        $data = Karyawan::where('kecamatan_id','=',$id)->get();
        return json_encode($data);
    }
    /**
     * Display the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
```

Lanjutan

```
    */
    public function show($id)
{
        //
    }

    /**
     * Show the form for editing the specified resource.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function edit($id)
    {
        //
    }
    /**
     * Update the specified resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @param  int  $id
     * @return \Illuminate\Http\Response
     */
    public function update(Request $request, $id)
    {
        //
    }
  /**
     * Remove the specified resource from storage.
     *
     * @param  int  $id
     * @return \Illuminate\Http\Response
```

```
     */
    public function destroy($id)
    {
        $data = Karyawan::findorfail($id)->delete();
        return redirect()->route('HomeKaryawan');
    }
}
```

10. Pembagian *Controller*

```
<?php
namespace App\Http\Controllers;
use App\MasterPembagian;
use Illuminate\Http\Request;
use App\Pembagian;
use App\Barang;
use App\Kecamatan;
use App\Karyawan;
use Auth;
use DB;
class PembagianController extends Controller
{
    public $dat = array();
    public $ank = 0;
    public function index()
    {
        $data = Pembagian::all();
        return view('pembagian',compact('data'));
    }
Lanjutan
    public function awal(){
        $data = Pembagian::all();
        return view('Pembagian2',compact('data'));
```

```
    }
public function awal3($id){
        $data   =   DB::select("select   date(created_at)   as   tanggal,
karyawan_id  FROM  `barang`  GROUP  BY  date(created_at),karyawan_id  having
karyawan_id = ".$id);
        // dd($data);
        return view('Pembagian2',compact('data'));
    }

    public function awal2($date){
        $kecamatan     =     DB::select("SELECT     b.karyawan_id     as
id,k.username,k.nama_lengkap,ke.kecamatan FROM `barang` b join karyawan k
on b.karyawan_id = k.id join kecamatan ke on ke.id = k.kecamatan_id where
date(created_at)          =          '2018-04-04'          GROUP          BY
b.karyawan_id,k.username,k.nama_lengkap,ke.kecamatan");
        // dd($kecamatan);
        return view('kecamatan',compact('date','kecamatan'));
    }

    public function test($date,$karyawan_id){
        $d     =     Barang::where(DB::raw('DATE(created_at)'),'=',$date)-
>where('karyawan_id','=',$karyawan_id)->orwhere('id','=',5)->get();
        // dd($d);
        $arr = json_decode($d);
        // dd($arr);
        $items = array();
        for ($i=0; $i < count($arr); $i++) {
            $items[] = $i;
}
        // dd($items);
        $data = $this->php_permutasi($items,array(),$d);
        // echo $this->ank;
        // dd($this->dat);
```

```
        $mine = $this->dat;
        // dd($mine);
        return view('hasil',compact('mine'));
    }

    public function php_permutasi($items, $perms = array( ),$d) {
        if (empty($items)) {
            // print $perms[0];
            // echo $d[0]->lang;
            $total = 0.0;
            $arr = array();
            for ($i=0; $i < count($perms)-1 ; $i++) {
                $subtotal=             $this->getDistance($d[$perms[$i]]-
>lat,$d[$perms[$i]]->lang,$d[$perms[$i+1]]->lat,$d[$perms[$i+1]]->lang);
                $total += $subtotal;
                $obj = array();
                $obj[] = $d[$perms[$i]]->nama_penerima;
                $obj[] = $d[$perms[$i]]->id;
                $obj[] = $d[$perms[$i+1]]->nama_penerima;
                $obj[] = $d[$perms[$i+1]]->id;
                $obj[] = $subtotal;
  $obj[] = $d[$perms[$i]]->lat;
                $obj[] = $d[$perms[$i]]->lang;
                $obj[] = $d[$perms[$i+1]]->lat;


    $obj[] = $d[$perms[$i+1]]->lang;
                $arr[] = $obj;
            }
            $this->ank++; //untuk testing
            $this->dat[] = $arr;
            // print join(' ', $perms) . ":".$total."<br/>";
        }  else {
```

```
            for ($i = count($items) - 1; $i >= 0; --$i) {
                $newitems = $items;
                $newperms = $perms;
                list($foo) = array_splice($newitems, $i, 1);
                array_unshift($newperms, $foo);
                $this->php_permutasi($newitems, $newperms,$d);
             }
        }
    }

    function    getDistance($point1_lat,    $point1_long,    $point2_lat,
$point2_long, $decimals = 2) {
            $degrees                                                    =
rad2deg(acos((sin(deg2rad($point1_lat))*sin(deg2rad($point2_lat)))      +
(cos(deg2rad($point1_lat))*cos(deg2rad($point2_lat))*cos(deg2rad($point1_
long-$point2_long))));
$distance = $degrees*111133.84;
            return round($distance, $decimals);
        }
    public function create($id)
    {
        return view('createPembagian',compact('id'));

   }
    public function createDetail($id){
        $barang = Barang::where('status','=','Baru')->get();
        return view('createDetailPembagian',compact('id','barang'));
    }
    public function store(Request $request,$id)
    {
        $data = new MasterPembagian;
        $data->tanggal = $request->tanggal;
        $data->karyawan_id = $id;
```

```
        $data->save();
        return redirect()->route('karyawanJob',['id' => $id]);
    }
    public function storeDetail(Request $request){
        $request->offsetUnset('_token');

Pembagian::create($request->all());
        return     redirect()->route('tambahDetailPembagian',['id'     =>
$request->master_pembagian_id]);
    }

    public function show($id)
    {
        $data = MasterPembagian::find($id);
        return view('detail_pembagian',compact('id','data'));
    }
    public function edit($id)
    {

    //
    }
    public function update(Request $request, $id)
    {
        //
    }
    public function destroy($id)
    {
        //
    }
    public function deleteDetail($id){
Pembagian::find($id)->delete();
        return back();
    }
```

```
}
```

11. *Model*

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Barang extends Model
{
    protected $table = 'barang';
    protected                          $fillable                          =
['nama_pengirim','nama_penerima','alamat','lang','lat','status','karyawan
_id'];
    }
```

12. Karyawan *Model*

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Karyawan extends Model
{
    protected $table = 'karyawan';
    public $timestamps = false;
    protected $hidden = [
        'password',
    ];
    function pembagian(){
       return $this->hasMany('App\MasterPembagian');
    }
    function kecamatan(){
       return $this->belongsTo('App\Kecamatan');
    }
}
```

13. Kecamatan *Model*

```
<?php
namespace App;
use Illuminate\Database\Eloquent\Model;
class Kecamatan extends Model
{
    protected $table = 'kecamatan';
    }
```

14. User *Model*

```
<?php
namespace App;
use Illuminate\Notifications\Notifiable;
use Illuminate\Foundation\Auth\User as Authenticatable;
class User extends Authenticatable
{
    use Notifiable;
    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name', 'email', 'password',
    ];
    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
```

```
     */
    protected $hidden = [
        'password', 'remember_token',
    ];
}
```