Aplikasi Rute Picking Order pada Gudang Menggunakan Metode Algoritma Harmony Search Berbasis Android

Karina Shelvi Gunawan, Andreas Handojo, Tanti Octavia
Program Studi Informatika Fakultas Teknologi Industri Universitas Kristen Petra
Jl. Siwalankerto 121 – 131 Surabaya 60236
Telp. (031) – 2983455, Fax. (031) – 8417658
E-Mail: karinasg12345@gmail.com, handojo@petra.ac.id, tanti@petra.ac.id

ABSTRAK

Gudang adalah suatu komponen yang penting dalam logistik. Gudang merupakan tempat penyimpanan barang atau bahan, baik berupa bahan baku (*raw material*), barang setengah jadi (*work-in-process*), atau barang jadi (*finished goods*). Gudang dimanfaatkan untuk menyimpan barang agar barang yang disimpan tidak cepat rusak. Teknologi dimanfaatkan untuk mempermudah dan membuat sistem pergudangan menjadi lebih efisien.

Penting bagi pemilik untuk mengetahui barang-barang yang ada pada gudang maupun pada toko retail. Barang yang masuk, keluar dan dimutasi harus diketahui agar tidak menyebabkan kerugian bagi pemilik. Untuk memenuhi pesanan dari konsumen maupun untuk menjaga jumlah stok barang pada rak di dalam toko retail, dilakukan *picking order* untuk mengambil barang-barang yang diperlukan dari dalam gudang. *Picking order* yang efisien dapat memenuhi pesanan lebih cepat sehingga dapat memenuhi lebih banyak pesanan yang datang.

Aplikasi ini memanfaatkan algoritma *Harmony Search* untuk melakukan perutean. Perutean dibuat untuk melakukan *picking order* pada gudang untuk memenuhi permintaan. Aplikasi juga dapat menampilkan visualisasi peta gudang untuk mempermudah pengguna. Selain itu, penggunaan sensor berat pada rak toko retail untuk mendeteksi stok pada rak toko retail. Pengguna dapat melakukan pemenuhan permintaan berdasarkan rute yang telah dihitung dengan algoritma *Harmony Search*.

Kata Kunci: *Harmony Search*, *Picking Order*, Rute, Sensor Berat, Pemenuhan Permintaan

ABSTRACT

Warehouse is an important element in logistic. Warehouse is place to keep raw material, work-in-process material, or finished goods. Warehouse is used to store goods so that it does not easily damaged. Technology is applied to make warehouse system more efficient.

It is important for the owner to know what goods are stored in the warehouse or retail store. Incoming, outgoing, and transferred goods must be known to prevent profit loss. To fulfill orders from consumers and to maintain the number of stock items on the shelves in retail store, picking orders are carried out to pick up the items needed from the warehouse. Efficient picking orders can fill orders faster and complete more incoming orders.

This application utilizes Harmony Search algorithm to do routing. Routing is made to do picking order at the warehouse to meet demand. The application can also display warehouse map visualizations to make it easier for users. In addition, the use of weight sensors on retail store shelves to detect stock on retail store shelves. Users can fulfill demands based on routes that have been calculated with the Harmony Search algorithm.

Keywords: *Harmony Search, Picking Order, Route, Weight Sensor, Demand Fulfillment*

1. PENDAHULUAN

Gudang merupakan komponen penting dalam sistem logistik. Gudang merupakan tempat penyimpanan barang atau bahan, baik berupa bahan baku (*raw material*), barang setengah jadi (*work-in-process*), atau barang jadi (*finished goods*) [6]. Pemantauan barang diperlukan untuk mengetahui kondisi *out-of-stock* (OOS) yaitu kasus dimana barang tidak tersedia untuk memenuhi permintaan pembeli sehingga menyebabkan kerugian [3]. Untuk memenuhi pesanan dari konsumen dan menjaga jumlah stok barang pada rak toko retail, dilakukan *picking order* untuk mengambil barang-barang yang diperlukan dari dalam gudang.

Picking order adalah sebuah proses mencari dan mengambil barang dari gudang untuk memenuhi pesanan pembeli [7]. *Picking order* yang efisien dapat memenuhi pesanan lebih cepat dan lebih banyak. Pekerja dapat menemui kesulitan dalam melakukan *picking order* karena harus mengingat letak setiap barang di dalam gudang. Hal tersebut dapat mengakibatkan keterlambatan dalam memenuhi pesanan pembeli. Pencarian rute dilakukan pada *picking order* agar mempermudah pekerja. Metode yang digunakan adalah algoritma *Harmony Search* (HS).

HS adalah sebuah algoritma optimasi metaheuristik berbasis populasi yang terinspirasi dari improvisasi musik [11]. Salah satu permasalahan yang dapat diselesaikan menggunakan HS adalah *Traveling Salesman Problem* (TSP). TSP adalah masalah pencarian rute terpendek mulai dari satu kota dan kembali ke kota yang sama, dengan hanya sekali melalui tiap kota dimana jarak antar kota diketahui [5]. Maka dari itu, HS merupakan metode algoritma yang cocok untuk digunakan dalam pencarian rute dalam *picking order*.

Aplikasi ini memungkinkan pengguna untuk mengetahui rute yang harus dilewati saat melakukan *picking order*. Sensor berat digunakan untuk mengetahui jumlah barang yang ada pada rak toko retail. Pengecekan rak toko retail dilakukan apabila rak telah mencapai batas minimum untuk dilakukan pengisian kembali.

2. DASAR TEORI

2.1 *Harmony Search*

Harmony Search (HS) adalah algoritma optimasi metaheuristik berbasis populasi [11]. Algoritma HS terinspirasi dari harmoni musik. Harmoni musik adalah kombinasi dari suara yang dianggap menyenangkan dari sudut pandang estetik [4]. Pertunjukan musik mencari kondisi terbaik yang ditentukan oleh estimasi estetik, dimana algoritma optimasi mencari kondisi

terbaik (*global optimum* – biaya minimal atau keuntungan maksimal atau efisiensi) yang ditentukan oleh evaluasi fungsi objektif.

Untuk mencari *global optimum*, HS menginisiasi sebuah parameter yaitu *Harmony Memory Considering Rate* (HMCR) yang memiliki *range* antara 0 sampai 1 [4]. Jika sebuah nilai yang dihasilkan seragam antara 0 sampai 1 di atas nilai HMCR sekarang, maka HS mencari not secara acak di dalam *range* yang dapat dimainkan tanpa memperdulikan HM. Untuk memperbaiki solusi dan keluar dari *local optima*, dimunculkan opsi lain yang meniru *pitch adjustment* dari tiap instrumen. Untuk komputasi, mekanisme *pitch adjustment* dirancang seperti berpindah ke *neighboring values* di dalam *range* dari nilai yang memungkinkan menggunakan *Pitch Adjusting Rate* (PAR).

Algoritma HS dijabarkan sebagai berikut [10]:

1. Menginisialisasi masalah dan algoritma
   Pertama, permasalahan optimasi adalah meminimalkan $f(x)$ dimana $Lx_i \leq x_i \leq Ux_i, \forall i \in \{1,2,\dots,N\}$. Parameter algoritma memasukan *harmony memory size* (HMS), HMCR, PAR, dan jumlah iterasi maksimum (T).

2. Menginisalisasi HM
   HM diisi dengan vektor solusi yang dihasilkan secara acak dan nilai fungsi yang berkaitan dihitung dan disimpan di dalam HM.

$$HM = \left(\begin{array}{ccccc|c} x_1^1 & x_2^1 & \cdots & x_{N-1}^1 & x_N^1 & f(\vec{x}^1) \\ x_1^2 & x_2^2 & \cdots & x_{N-1}^2 & x_N^2 & f(\vec{x}^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ x_1^{HMS-1} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS-1} & x_N^{HMS-1} & f(\vec{x}^{HMS-1}) \\ x_1^{HMS} & x_2^{HMS-1} & \cdots & x_{N-1}^{HMS} & x_N^{HMS} & f(\vec{x}^{HMS}) \end{array}\right)$$

3. Mengimprovisasi harmony memory
   Harmoni baru dinyatakan sebagai $x' = [x_1' x_2' \dots x_{N-1}' x_N']$ diimprovisasi menggunakan tiga rule: *memory consideration rule*, *pitch adjustment rule*, dan *randomization*. Algoritma mengeksekusinya sebagai berikut: (i) menghasilkan solusi baru dari HM (*memory consideration*); (ii) mengganti variabel keputusan dengan variabel baru yang mendekati variabel saat ini (*pitch adjustment*); (iii) menghasilkan vektor solusi dari *random range* yang memungkinkan (*random selection*). *Memory consideration* dan *random selection* dilakukan dengan menggunakan HMCR yang ditentukan sebelumnya,

$$If\ rand(0,1) < HMCR\ then$$
$$x_i' \leftarrow x_i' \in \{x_i^1, x_i^2, \dots, x_i^{HMS}\}\ Else$$
$$x_i' \leftarrow x_i' \in \{x_{i,min}, x_{i,max}\}\ End\ If$$

   dimana $random(0,1)$ adalah angka acak yang seragam antara 0 dan 1.

   Variabel keputusan baru diperlukan untuk menentukan apakah *pitch adjustment* diperlukan atau tidak. Menggunakan nilai PAR, *pitch adjustment* dilakukan seperti berikut:

$$If\ rand(0,1) < PAR\ then$$
$$x_i' \leftarrow x_i' \pm rand(0,1) \times bw\ Else$$
$$x_i' \leftarrow x_i'\ End\ If$$

   dimana *bw* adalah *bandwidth* yang digunakan untuk *pitch adjustment*.

4. Memperbarui harmony memory
   Harmoni baru, $x' = [x_1' x_2' \dots x_{N-1}' x_N']$, dan harmoni terburuk di HM dibandingkan dalam hal *fitness values*. Harmoni yang lebih baik dimasukkan ke dalam HM sedangkan yang terburuk dihilangkan.

5. Memeriksa kriteria penghentian (*termination*)
   Iterasi dihitung sejak langkah ketiga hingga langkah kelima. Kriteria penghentian diperiksa pada langkah kelima.

2.2 *Picking Order*

Picking order merupakan kegiatan mengumpulkan barang-barang dari penyimpanan dalam jumlah tertentu dari pesanan yang diberikan oleh pembeli [9]. Metode-metode *picking order* dibagi berdasarkan siapa yang mengambil pesanan (manusia atau mesin), siapa yang bergerak (pengambil barang atau barangnya) dan strategi yang digunakan dalam mengambil pesanan tersebut.

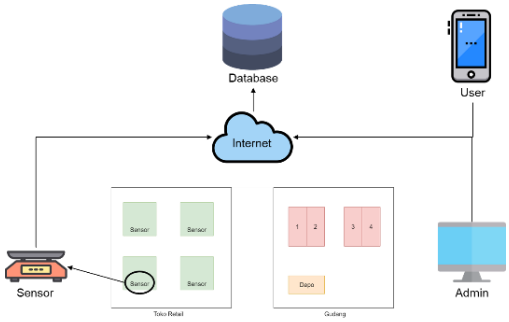
2.3 *Load Cell*

Load cell adalah sebuah elemen fisik (*transducer*) yang menerjemahkan tekanan menjadi sebuah sinyal elektrik [2]. Terdapat tiga cara bagi *load cell* untuk menerjemahkan tekanan menjadi sebuah pembacaan yang dapat diukur yaitu *Hydraulic Load Cell*, *Pneumatic Load Cell*, dan *Strain Gauge Load Cell*.

3. DESAIN SISTEM

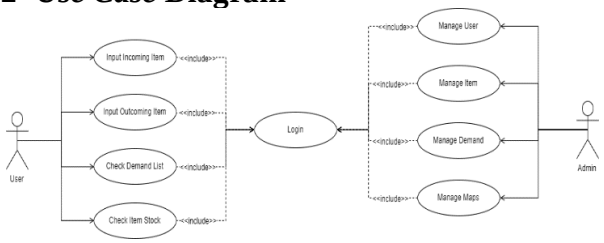
3.1 Desain Sistem Umum

Desain sistem dapat dilihat pada Gambar 1. *User* menggunakan aplikasi *Android*, dan admin menggunakan *web*. *User* dapat memasukkan permintaan barang yang masuk dan keluar dari gudang dan dapat mengecek jumlah stok yang ada pada gudang dan toko retail. Admin dapat mengatur *user*, permintaan, barang-barang dan peta gudang melalui *web*. Sensor berat yang dipasang pada rak toko retail digunakan untuk memberitahu apabila stok barang mencapai batas minimum, sehingga membuat permintaan berdasarkan sensor tersebut.



Gambar 1. Desain Sistem Umum

3.2 *Use Case* Diagram



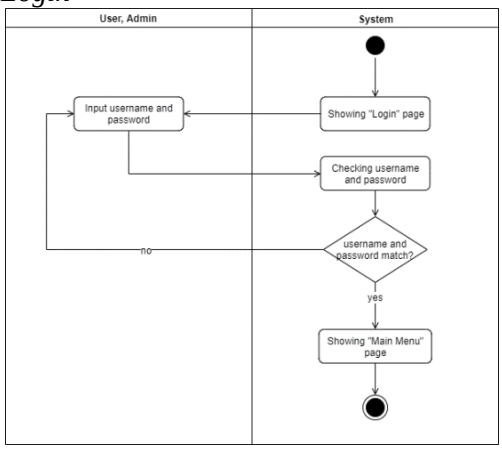
Gambar 2. *Use Case Diagram*

Use case diagram untuk aplikasi ini dapat dilihat melalui Gambar 2. *User* dapat melakukan *input incoming item*, *input outcoming item*, *check demand list* dan *check item stock*. *Input incoming item*

merupakan proses penginputan barang yang masuk ke dalam gudang. *Input outcoming item* adalah penginputan barang yang keluar dari gudang. Sedangkan *check detail list* untuk mengecek daftar permintaan yang ada untuk diselesaikan dan *check item stock* merupakan proses untuk mengecek stok yang berada pada gudang dan toko retail. Admin dapat melakukan *manage user*, *manage item*, *manage demand*, dan *manage maps*. Setiap pengaturan admin terdiri dari *insert*, *edit*, dan *delete*. *Insert* merupakan proses memasukkan data baru ke dalam *database*. *Edit* adalah proses menyunting data yang sudah ada di dalam *database*. Sedangkan *delete* adalah penghapusan data yang ada di dalam *database*.

3.3 *Activity Diagram*

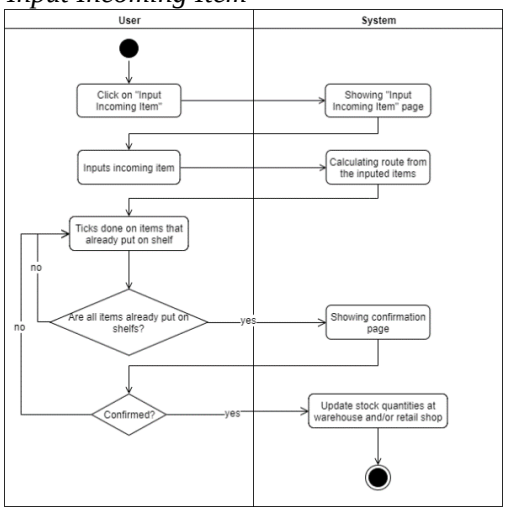
3.3.1 Login



Gambar 3. ***Activity Diagram Login***

Activity diagram untuk *login* dapat dilihat pada Gambar 3. Sistem akan menampilkan halaman *Login*. *User* atau admin memasukkan *username* dan *password* pada halaman *Login* tersebut. Kemudian sistem akan mengecek apakah *username* dan *password* yang dimasukkan cocok dengan data yang ada pada *database*. Jika *username* dan *password* cocok, maka *user* atau admin berhasil *login*. Sedangkan apabila tidak cocok, *user* atau admin diminta untuk memasukkan lagi *username* dan *password*.

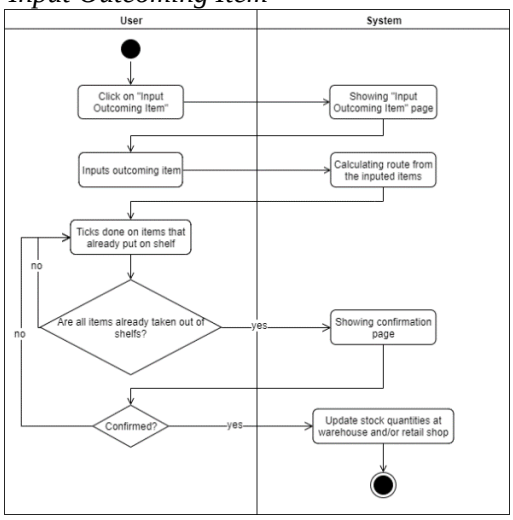
3.3.2 Input Incoming Item



Gambar 4. ***Activity Diagram Input Incoming Item***

Activity diagram dapat dilihat pada Gambar 4. Pertama, *user* mengklik menu *Input Incoming Item*. Sistem menampilkan halaman untuk menu *Input Incoming Item* dan *user* memasukkan data barang-barang yang masuk ke dalam gudang. Setelah itu, sistem akan melakukan penghitungan rute dari data barang yang telah dimasukkan menggunakan algoritma *Harmony Search*. *User* mencentang barang yang telah dimasukkan ke dalam rak. Apabila semua barang telah dicentang, sistem memunculkan halaman konfirmasi sebelum memasukkan *list* ke dalam *database*. Setelah dikonfirmasi, maka sistem memperbarui jumlah stok pada gudang dan/atau toko retail.

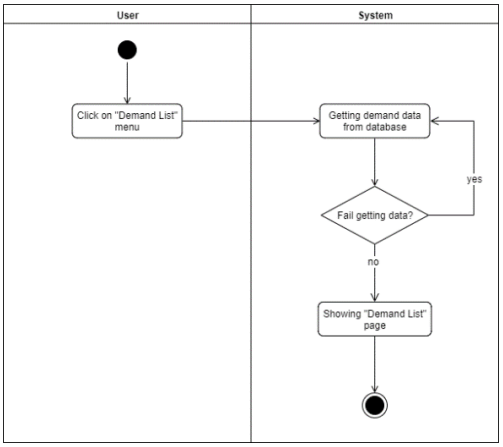
3.3.3 Input Outcoming Item



Gambar 5. ***Activity Diagram Input Outcoming Item***

Gambar 5 menunjukkan *activity diagram* untuk *input outcoming item*. Pertama, *user* mengklik menu *Input Outcoming Item* dan sistem menampilkan halaman untuk menu *Input Outcoming Item*. *User* memasukkan data barang yang keluar dari gudang. Setelah itu, sistem melakukan penghitungan rute dari data barang yang telah dimasukkan menggunakan algoritma *Harmony Search*. *User* akan mencentang barang yang telah dikeluarkan dari dalam rak. Apabila semua barang telah dicentang, sistem memunculkan halaman konfirmasi sebelum memasukkan *list* ke dalam *database*. Setelah dikonfirmasi, sistem memperbarui jumlah stok pada gudang dan/atau toko retail.

3.3.4 Check Demand List

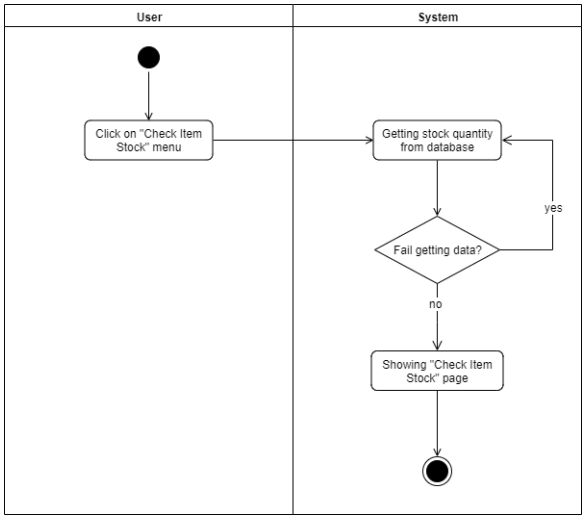


Gambar 6. ***Activity Diagram Check Demand List***

Activity diagram untuk *check demand list* dapat dilihat pada Gambar 6. Pertama, *user* mengklik menu *Demand List*. Sistem akan mengambil data permintaan barang dari *database*. Apabila

sistem berhasil mengambil data, sistem menampilkan halaman *Demand List*. Jika tidak berhasil, maka sistem akan mengambil data lagi.

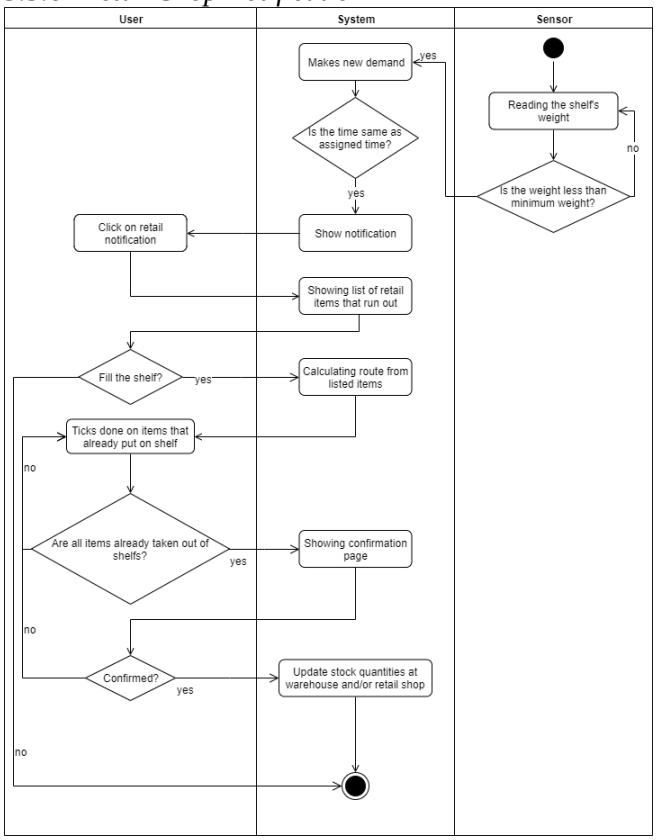
3.3.5 Check Item Stock



Gambar 7. *Activity Diagram Check Item Stock*

Gambar 7 adalah *activity diagram* untuk *check item stock*. *User* akan mengklik menu *Check Item Stock*. Lalu, sistem akan mengambil data jumlah stok gudang dan toko retail dari *database*. Apabila sistem berhasil mengambil data, sistem akan menampilkan halaman *Check Item Stock*. Jika tidak berhasil, maka sistem akan mengambil data lagi.

3.3.6 Retail Shop Notification



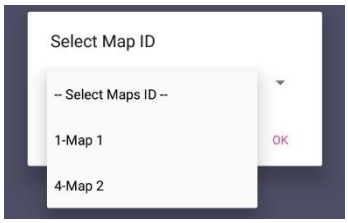
Gambar 8. *Activity Diagram Retail Shop Notification*

Activity diagram retail shop notification dapat dilihat pada Gambar 8. Sensor berat akan mendeteksi berat dari rak toko retail untuk mengidentifikasi jumlah barang yang ada pada rak toko retail. Apabila berat telah mencapai batas minimum, sistem membuat permintaan baru ke dalam *database*. Pada jam yang ditentukan, aplikasi memunculkan notifikasi ke aplikasi. *User* akan mengklik notifikasi tersebut dan sistem memunculkan *list* barang-barang yang akan diisi raknya. Apabila *user* memutuskan untuk mengisi rak toko retail, sistem menghitung rute untuk melakukan pengambilan barang pada gudang. Apabila barang telah diambil seluruhnya, sistem menampilkan halaman konfirmasi. Setelah dikonfirmasi, sistem memperbarui jumlah stok pada *database*.

4. PENGUJIAN SISTEM

4.1 Pengujian Aplikasi

Sebelum benar-benar masuk Halaman *Input Incoming Item*, *user* akan ditampilkan sebuah kotak dialog seperti pada Gambar 9. *User* dapat memilih peta gudang mana yang akan dipilih untuk mengambil barang. Halaman *Input Incoming Item* pada Gambar 10 menampilkan dua kolom untuk memasukkan nama barang dan jumlah barang. Ketika tombol *Input* ditekan, barang yang telah dimasukkan akan dipindahkan ke dalam tabel. Tombol *Calculate Route* akan memasukkan semua permintaan dari dalam tabel ke *database* lalu menampilkan halaman *Route*.

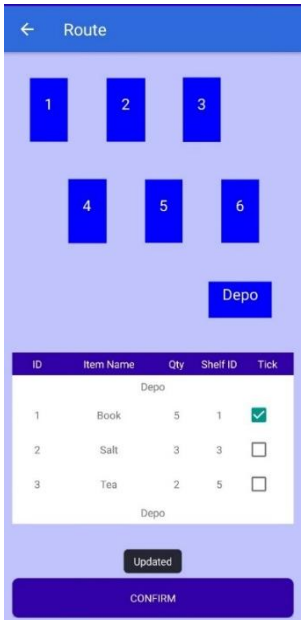


Gambar 9. Kotak Dialog *Maps*

Gambar 10. Halaman *Input Incoming Item*

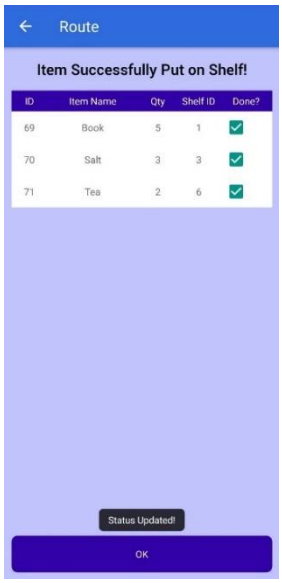
Halaman *Route* (Gambar 11) menampilkan gambar peta gudang dimana permintaan tersebut dikerjakan. Halaman *Route* juga menampilkan tabel yang berisi detail permintaan yang sudah

diurutkan berdasarkan hasil perhitungan algoritma HS. Pada tiap baris dalam tabel diberi *checkbox*. *User* akan mencentang *checkbox* apabila telah menyelesaikan detail permintaan. Tombol *Confirm* ditekan ketika seluruh detail permintaan telah dikerjakan.



Gambar 11. Halaman *Route*

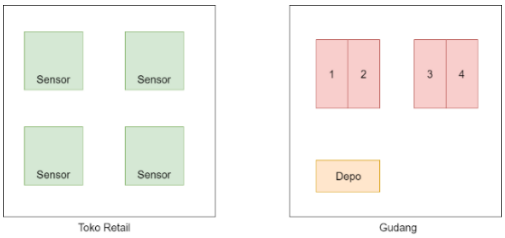
Apabila semua detail permintaan telah dikerjakan, aplikasi akan mengarahkan *user* ke halaman Konfirmasi. Halaman Konfirmasi dapat dilihat pada Gambar 12.



Gambar 12. Halaman Konfirmasi

4.2 Model Skenario

Pembuatan model skenario digunakan untuk memberikan gambaran mengenai sistem pada gudang dan toko retail. Model skenario dapat dilihat pada Gambar 13.



Gambar 13. Skenario Sistem Gudang dan Toko Retail

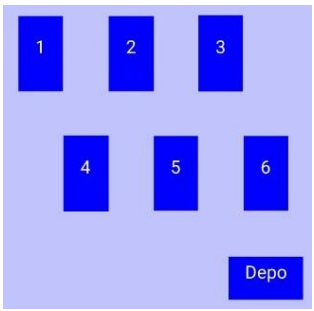
Toko retail memiliki rak yang dipasang sensor berat. Sensor berat pada rak berfungsi untuk mendeteksi apabila barang pada rak telah mencapai batas minimum. Ketika telah mencapai batas minimum, sistem akan membuat detail permintaan baru yang berisi barang dari rak toko retail yang harus diisi. *User* akan diberi notifikasi pada waktu tertentu untuk mengerjakan permintaan dari toko retail. Permintaan juga dapat dibuat secara manual oleh *user* melalui aplikasi.

Jika *user* menerima permintaan, aplikasi akan menentukan urutan rak-rak yang dikunjungi menggunakan algoritma HS. Kapasitas yang dapat ditampung oleh pengangkut barang sebesar 10kg. Apabila pengangkut barang penuh, *user* harus kembali ke depo untuk menurunkan muatannya. Setiap rak pada gudang berisi satu jenis barang. Penentuan urutan rak-rak yang dikunjungi berdasarkan jarak tempuh terpendek.Setelah permintaan diselesaikan, sistem akan mengurangi atau menambahkan stok pada gudang. Stok pada toko retail juga akan bertambah saat *user* menyelesaikan permintaan dari toko retail.

4.3 Verifikasi Model

Verifikasi model dilakukan untuk mengetahui apakah model yang dibuat sudah sesuai dengan keadaan sebenarnya. Permintaan yang ada akan dihitung jarak optimal antar raknya menggunakan aplikasi dan perhitungan manual.

Gambar 14 adalah peta gudang. Jarak antar rak pada gudang dapat dilihat pada Tabel 1. Rak yang tersedia diberi angka 1-6 dan untuk depo diberi angka 0. Tabel 2 menunjukkan model permintaan yang akan dikerjakan pada aplikasi.



Gambar 14. Peta Gudang

Tabel 1. Jarak Antar Rak

Rak	0	1	2	3	4	5	6
0	0	25	20	15	15	10	5
1	25	0	5	20	10	10	15
2	1	5	0	5	10	10	10
3	15	20	5	0	15	10	10
4	15	10	20	15	0	5	20
5	10	10	10	10	5	0	5
6	5	15	10	10	20	5	0

Tabel 2. Model Permintaan

item	qty	weight	shelf	total weight
milk	10	250	1	2500
soy sauce	5	100	2	500
salt	25	50	3	1250
water bottle	5	250	4	1250
tea	20	100	5	2000
pen	15	50	6	750
				8250

Model permintaan selanjutnya diproses dalam aplikasi menggunakan algoritma HS. Jumlah iterasi yang ditetapkan dalam uji coba ini sebanyak 1 kali. Parameter HMCR dan PAR didapatkan dari penelitian Octavia dan Angel [8]. Jarak yang dihasilkan adalah sebesar 70 meter. Hasil yang diperoleh oleh aplikasi dapat dilihat pada Tabel 3.

Tabel 3. Hasil Akhir dari Aplikasi

Rute Rak									Jarak
V1	0	6	2	3	5	4	1	0	70

4.4 Uji Coba CPU *Time*

Pengujian CPU *time* adalah pengujian perhitungan waktu yang diperlukan aplikasi untuk mendapatkan rute optimal. Aplikasi akan menjalankan sebanyak 5, 10, dan 15 permintaan. Aplikasi dijalankan menggunakan Oppo F9 dengan sistem operasi Android 9 Processor Octa Core dan RAM 4GB dengan memori internal 64GB. Tabel 4 menunjukkan CPU *time* yang dibutuhkan untuk menjalankan algoritma HS.

Tabel 4. CPU *Time*

Permintaan	CPU Time (ms)					Rata-Rata
	I	II	III	IV	V	
5 Rak	552	435	483	496	500	493,2
10 Rak	569	595	594	606	661	605
15 Rak	890	891	887	842	826	867,2
20 Rak	1039	1074	1037	1027	1036	1042,6
					Rata-Rata :	752

5. KESIMPULAN

Berdasarkan hasil pengujian yang telah dilakukan, dapat diambil beberapa kesimpulan antara lain :

- Aplikasi rute *picking order* dapat memudahkan *user* dalam menyelesaikan permintaan.
- Pengaplikasian metode algoritma HS dapat mencari rute terpendek sesuai permintaan.
- Jumlah iterasi maksimal yang dibutuhkan sebesar 792 kali.
- Semakin banyak permintaan, semakin banyak iterasi yang diperlukan untuk mencapai jarak tempuh terbaik.
- Penurunan nilai terhadap jarak awal dan jarak akhir rata-ratanya adalah 28,76%.
- Waktu yang diperlukan untuk mencari rute rata-ratanya adalah 752ms.
- Sensor berat dapat digunakan untuk mendeteksi berat barang pada rak toko retail.
- Aplikasi dapat mengirimkan notifikasi pada waktu tertentu untuk mengingatkan *user* mengisi rak toko retail yang beratnya di bawah batas minimum.

Saran untuk pengembangan kedepannnya adalah:

- Meningkatkan tampilan *User Interface* aplikasi dan *website.*
- Melakukan visualisasi rute yang dilewati.
- Algoritma HS juga dapat dijalankan pada admin.
- Mempertimbangkan parameter berat dan volume barang dalam perhitungan algoritma HS.
- Sensor dapat mendeteksi apabila barang diletakkan di rak yang salah.
- Perhitungan jarak antar rak pada peta dapat dilakukan secara otomatis.

6. DAFTAR PUSTAKA

[1] Alia, O., dan Mandava, R. 2011. The variants of the harmony search algorithm: an overview. *Artificial Intellegence Review*. DOI= https://doi.org/10.1007/s10462-010-9201-y.

[2] Al-Mutlaq, S. n.d. *Getting Started with Load Cells*. URI= https://learn.sparkfun.com/tutorials/getting-started-with-load-cells/all.

[3] Frontoni, E., Marinelli, F., Rosetti, R., dan Zingaretti, P. 2017. Shelf space re-allocation for out of stock reduction. *Computers and Industrial Engineering, 106*, 32-40. DOI= https://doi.org/10.1016/j.cie.2017.01.021.

[4] Geem, Z., Kim, J., dan Loganathan, G. 2001. A New Heuristic Optimization Algorithm : Harmony Search. 60-68. DOI= https://doi.org/10.1177/003754970107600201.

[5] Hacizade, U., dan Kaya, I. 2018. GA Based Traveling Salesman Problem Solution and its Application to Transport Routes Optimization. *IFAC-PapersOnLine*, *51*(30), 620-625. DOI= https://doi.org/10.1016/j.ifacol.2018.11.224.

[6] Lestari, S. 2016. *Fungsi Gudang dalam Sistem Logistik dan Rantai Pasok*. URI= http://supplychainindonesia.com/new/fungsi-gudang-dalam-sistem-logistik-dan-rantai-pasok/.

[7] Murray, M. 2019. *Order Picking in Warehouse*. URI= https://www.thebalancesmb.com/order-picking-in-the-warehouse-2221190.

[8] Octavia, T., dan Angelica, S. 2017. Perbandingan Algoritma Simulated Annealing dan Harmony Search dalam Penerapan Picking Order Sequence. *Jurnal Teknik Industri*, 125-132. DOI= https://doi.org/10.9744/jti.19.2.125-132.

[9] Pansart, L., Catusse, N., dan Cambazard, H. 2018. Exact algorithms for the order picking problem. *Computers and Operations Research, 100*, 117-127. DOI= https://doi.org/10.1016/j.cor.2018.07.002.

[10] Tongchan, T., Pomsing, C., dan Tonglim, T. 2017. Harmony Search Algorithm's Parameter Tuning for Traveling Salesman Problem. *2017 International Conference on Robotics and Automation Sciences, ICRAS 2017*, 199-203. DOI= https://doi.org/10.1109/ICRAS.2017.8071944.

[11] Zhang, T., dan Geem, Z. 2019. Review of harmony search with respect to algorithm structure. *Swarm and Evolutionary Computation*, 31-43. DOI= https://doi.org/10.1016/j.swevo.2019.03.012.