

LAPORAN PROJECT AKHIR
ALGORITMA DAN PEMROGRAMAN II



PERANCANGAN APLIKASI GO CARGO SEBAGAI SOLUSI
OPTIMALISASI MUAT BARANG DENGAN PENERAPAN
ALGORITMA KNAPSACK PROBLEM

Dosen Pengampu:

Priza Pandunata, S.Kom., M.Sc.

Mohammad Zarkasi, S.Kom., M.Kom.

Tio Dharmawan, S.Kom., M.Kom.

Disusun Oleh :

Nur Muhammad Himawan (202410101070)

Varrel Dwantio P. (202410101081)

FAKULTAS ILMU KOMPUTER
UNIVERSITAS JEMBER
TAHUN AKADEMIK 2020/2021 GENAP

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan ke hadirat Allah SWT karena atas segala karunia- Nya penulis dapat menyelesaikan penyusunan laporan tugas akhir ini. Penyusunan laporan ini dibuat untuk memenuhi evaluasi tugas akhir mata kuliah Algoritma dan Pemrograman I.

Dalam usaha menyelesaikan laporan tugas akhir ini penulis sudah banyak mendapatkan bantuan maupun dorongan dari para dosen, asdos serta pihak lain dan teman-teman seperjuangan. Untuk itu pada kesempatan ini penulis mengucapkan terima kasih yang sedalam-dalamnya kepada yang terhormat:

1. Priza Pandunata. S.Kom., M.Sc., Mohammad Zarkasi, S.Kom., M.Kom., Tio Dharmawan, S.Kom., M.Kom., selaku Dosen mata kuliah Algoritma dan Pemrograman II yang telah memberikan materi, penjelasan dan pengarahan yang berguna untuk pembuatan progam serta penyelesain laporan tugas akhir ini.
2. Handika Catur Pratama, Chinta ‘Aliyyah Candramaya, dkk. selaku Asisten Dosen mata kuliah Algoritma dan Pemrograman II yang telah memberikan materi, masukan dan saran untuk kemajuan penyelesaian progam, penyusunan laporan ini serta membimbing selama proses perkuliahan.
3. Orang tua dan kakak-kakak penulis, yang selalu mendo’akan kelancaran penyelesaian laporan tugas akhir ini dan memberikan dukungan moral dan materi.
4. Teman-teman seperjuangan Fakultas Ilmu Komputer Universitas Jember 2020/2021 yang saling membantu saat ada kesulitan dan saling bertukar pikiran tentang penyelesaian tugas dan penyusunan laporan ini.

Semoga laporan tugas akhir ini dapat memberikan wawasan yang lebih luas dan menjadi sumbangan pemikiran kepada pembaca. Penulis menyadari bahwa dalam penyusunan laporan ini jauh dari kata sempurna, baik dari segi penyusunan, bahasan, penulisan, ataupun hasil akhir perancangan progam. Oleh karena itu kami mengharapkan kritik dan saran yang sifatnya membangun, untuk menjadi acuan dalam bekal pengalaman bagi penulis untuk lebih baik di masa yang akan datang.

Jember, 12 Juni 2021

Penulis

DAFTAR ISI

KATA PENGANTAR	i
DAFTAR ISI.....	ii
BAB I. DASAR TEORI.....	1
1.1. Pengertian Algoritma dan Kode Progam	1
1.2. Knapsack Problem	3
1.3. Spanning Tree	5
1.4. Maze Problem	7
1.5. Shortest Path Problem.....	8
1.6. Travelling Salesmen Problem	10
BAB II. TOPIK PROJECT	12
2.1. Topik Project.....	12
2.2. Tujuan Pembuatan Aplikasi	14
2.3. Batasan Aplikasi	14
BAB III. RANCANGAN APLIKASI.....	15
3.1. Kebutuhan Sistem	15
3.2. Struktur Data.....	15
3.3. Rancangan Langkah Penggunaan Aplikasi	17
BAB IV. IMPLEMENTASI RANCANGAN	19
4.1. Import Module	19
4.2. Autentikasi	19
4.3. Fitur Login	22
4.4. Fitur Registrasi.....	23
4.5. Sidebar	24
4.6. Menu Pelanggan.....	28
4.6.1. Fitur Home	28
4.6.2. Fitur Lihat Pengiriman	31
4.6.3. Fitur Lihat Profil.....	32
4.7. Menu Pengirim.....	33
4.7.1. Fitur Home	34
4.7.2. Fitur Lihat Penyimpanan Barang	38
4.7.3. Fitur Lihat Profil	40
4.8. Looping	42
BAB V. HASIL DAN PEMBAHASAN.....	43

5.1.	Untuk Users sebagai Customers	43
5.1.1.	Menu Login.....	43
5.1.2.	Menu Registerasi Akun.....	44
5.1.3.	Menu Home atau Beranda.....	45
5.1.4.	Menu Cek Pengiriman	46
5.1.5.	Menu Profile	47
5.1.6.	Exit.....	47
5.2.	Untuk Users sebagai Admin Pengirim.....	47
5.2.1.	Menu Home atau Beranda	47
5.2.2.	Menu Storage	49
5.2.3.	Menu Profile.....	49
5.2.4.	Exit	50
DAFTAR PUSTAKA		51
LAMPIRAN.....		52

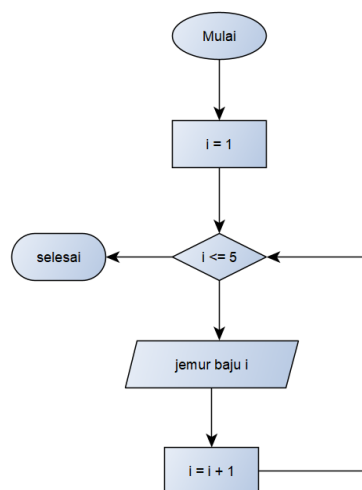
BAB I

DASAR TEORI

1.1. Pengertian Algoritma dan Kode Program

Algoritma adalah prosedur yang berisi langkah-langkah untuk sebuah perhitungan dalam menyelesaikan sebuah masalah yang ditulis dalam sebuah urutan. Sedangkan, kode program adalah suatu rangkaian pernyataan atau deklarasi yang ditulis dalam bahasa pemrograman komputer yang terbaca manusia.

- Perulangan atau Looping

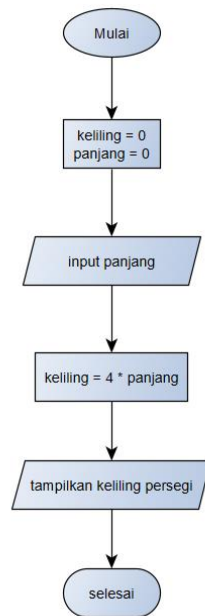


Gambar 1.1-1. Flowchart Perulangan

Perulangan atau looping dalam algoritma merupakan proses menjalankan beberapa operasi tertentu secara berulang-ulang. Algoritma perulangan ini bisa dianalogikan ketika dalam proses menjemur pakain. Dalam proses tersebut terdapat pengulangan dari setiap baju yang akan dijemur. Contoh kode program dari gambar 1.1-1. sebagai berikut :

```
for i in range(1, 6):
    print(f"Jemur baju {i}")
    i += 1
```

- Sekuensial

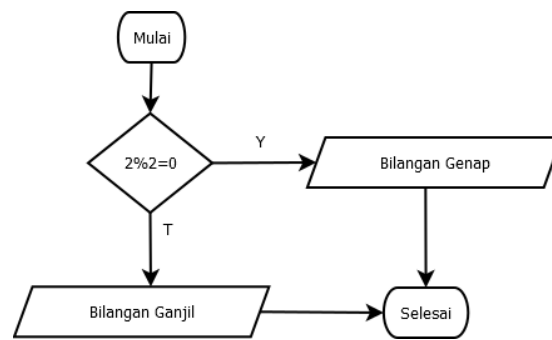


Gambar 1.1.-2. Flowchart Sekuensial

Sekuensial merupakan algoritma yang langkah-langkahnya urut dari awal sampai akhir. Salah satu contoh dari sekuensial dalam kehidupan sehari-hari yaitu memasak mie instan. Langkah-langkah memasak mie instan harus urut dari atas sampai bawah. Berikut contoh kode program sesuai gambar 1.1-2 :

```
keliling = 0
panjang = 0
panjang = int(input("Masukkan Panjang: "))
keliling = 4 * panjang
print(f"Keliling Persegi = {keliling} cm")
```

- Percabangan atau Bersyarat



Gambar 1.1-3. Flowchart Percabangan

Percabangan merupakan algoritma yang akan menjalankan suatu operasi jika syarat dari operasi yang dijalankan terpenuhi. Contoh kode program dari gambar 1.1-3 sebagai berikut :

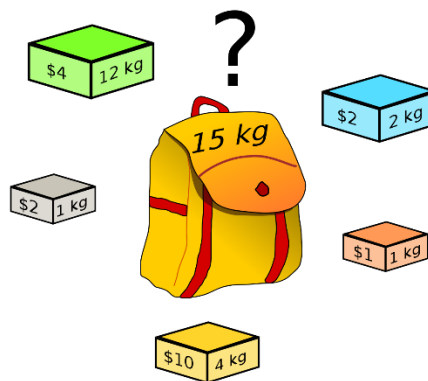
```
if 2 % 2 == 0:
    print("Bilangan Genap")
else:
    print("Bilangan Ganjil")
```

1.2. Knapsack Problem

Knapsack problem merupakan masalah di mana orang dihadapkan pada persoalan optimasi pada pemilihan benda yang dapat dimasukkan ke dalam sebuah wadah yang memiliki keterbatasan ruang atau daya tampung. Dengan adanya optimasi dalam pemilihan benda yang akan dimasukkan ke dalam wadah tersebut diharapkan dapat menghasilkan keuntungan yang maksimum.

Benda-benda yang akan dimasukkan ini masing-masing memiliki berat dan sebuah nilai yang digunakan untuk menentukan prioritasnya dalam pemilihan tersebut. Nilainya dapat berupa tingkat kepentingan, harga barang, nilai sejarah, atau yang lainnya. Wadah yang dimaksud di sini juga memiliki nilai konstanta yang merupakan nilai pembatas untuk benda-benda yang akan dimasukkan ke dalam wadah tersebut sehingga harus diambil sebuah cara memasukkan benda-benda tersebut ke dalam wadah sehingga menghasilkan hasil optimum tetapi tidak melebihi kemampuan wadah untuk menampungnya. Knapsack problem sendiri memiliki beberapa jenis antaralain :

- 0/1 Knapsack problem yaitu tiap barang hanya tersedia sebanyak 1 unit. Jika sesuai dengan keuntungan maksimal maka dipilih barang tersebut, sedangkan jika tidak maka dilepaskan atau dibiarkan.
- Fraksional knapsack problem yaitu barang yang bisa dibawa hanya sebagian. Jenis problem ini bisa masuk akal jika barang yang ada bisa dibagi seperti tepung, gula dan lain-lain.
- Bounded knapsack problem yaitu masing-masing barang tersedia dalam N unit yang mana jumlahnya terbatas.
- Unbounded knapsack problem yaitu masing-masing barang yang tersedia jumlahnya minimal dua unit atau bahkan tak terbatas.



Gambar 1.2-1. Knapsack Problem

Pada gambar 1.2-1 terdapat 5 kotak yang memiliki berat dan harga yang berbeda. Dari kelima kotak tersebut dipilih yang nantinya akan dimasukkan ke dalam tas. Tetapi, tas hanya menampung total berat kotak kurang dari 15kg. Pemilihan kotak harus berdasarkan jumlah keuntungan yang paling besar dengan memperhatikan berat kotaknya sehingga tidak melebihi maksimal daya tampung tas.

Terdapat 2 algoritma yang dapat digunakan dalam mengatasi knapsack problem yaitu Brute Force Algorithm dan Greedy Algorithm. Brute force sendiri merupakan pemecahan masalah dengan teknik straightforward sehingga dapat diperoleh sebuah keputusan pemecahan masalah yang langsung mengacu atau menuju kepada hasil yang diinginkan. Sedangkan, greedy algorithm merupakan metode yang paling populer untuk memecahkan persoalan optimasi. Prinsip dari greedy algorithm yaitu mengambil apa yang bisa didapatkan sekarang. Sesuai gambar 1.2-1 bisa diatasi dengan Greedy Algorithm. Greedy memiliki 3 pilihan strategi pengangkutan, yaitu:

- ***Greedy by Profit***

Strategi ini mencoba memaksimumkan keuntungan dengan memilih objek yang paling menguntungkan terlebih dahulu. Pertama kali dilakukan adalah mengurutkan secara menurun obyek-obyek berdasarkan profitnya. Kemudian obyek-obyek yang dapat ditampung oleh knapsack diambil satu persatu sampai knapsack penuh atau (sudah tidak ada obyek lagi yang bisa dimasukan). Sesuai dengan gambar 1.1-1. maka didapatkan data awal sebagai berikut :

1. $W_1 = 4 \text{ kg}; P_1 = \10

2. $W_2 = 12 \text{ kg}; P_2 = \4

3. $W_3 = 1 \text{ kg}; P_3 = \2

4. $W_4 = 2 \text{ kg}; P_4 = \2

5. $W_5 = 1 \text{ kg}; P_5 = \1

Kapasitas tas (W_{\max}) = 15 kg

- ***Greedy by Weight***

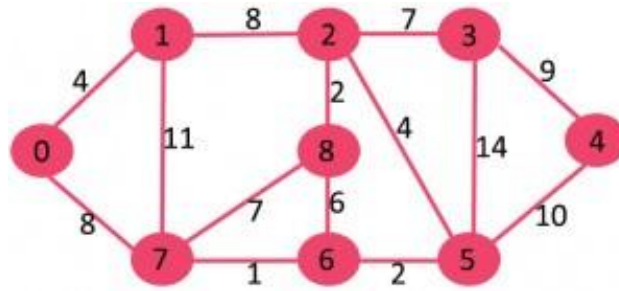
Pada strategi ini fokus dengan memasukan barang sebanyak-banyaknya kedalam tas atau kantong. Jadi barang atau objek yang dimasukan terlebih dahulu adalah barang dengan bobot paling ringan terlebih dahulu, dengan harapan dengan banyaknya barang atau objek yang terangkut kita bisa mendapatkan keuntungan sebesar-besarnya.

- ***Greedy by Density***

Strategi greedy by density berfokus untuk mendapat keuntungan sebesar-besarnya dengan memasukan barang yang memiliki keuntungan per unit terbesar (P_i/W_i) terlebih dahulu kedalam kantong.

1.3. Spanning Tree

Spanning tree merupakan suatu graf berbobot tak berarah yang menghubungkan semua titik tanpa membentuk siklus dan dengan total bobot minimum.



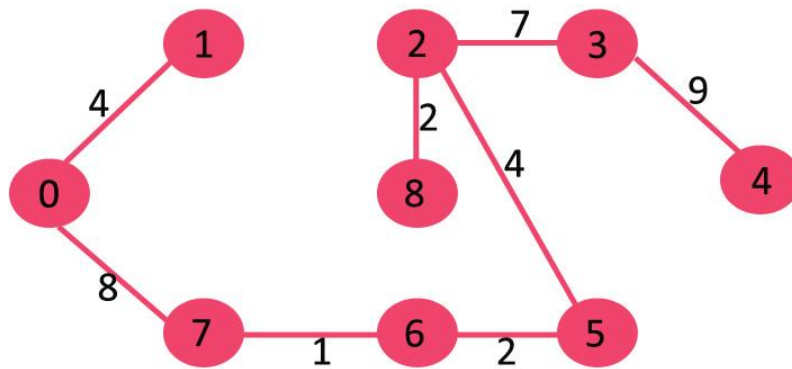
Gambar 1.3-1. Graph

Pada gambar 1.3-1 terdapat graph berisi 9 simpul dan 14 sisi. Jadi, pohon merentang minimum yang terbentuk akan memiliki $(9 - 1) = 8$ tepi. Jadi bisa diurutkan dimulai dari bobot yang paling rendah sesuai tabel dibawah ini.

s	Sumber	Tujuan
1	7	6
2	8	2
2	6	5
4	0	1
4	2	5
6	8	6
7	2	3
7	7	8
8	0	7
8	1	2
9	3	4
10	5	4
11	1	7
14	3	5

Tabel 1.3-1. Urutan Bobot Graph

Setelah memilih semua tepi satu per satu dari daftar tepi yang diurutkan di table 1.3-1 dengan pertimbangan tidak membentuk sebuah siklus. Algoritma akan berhenti Ketika $(V - 1)$ yang mana V adalah jumlah simpul maka diperoleh hasil sebagai berikut :

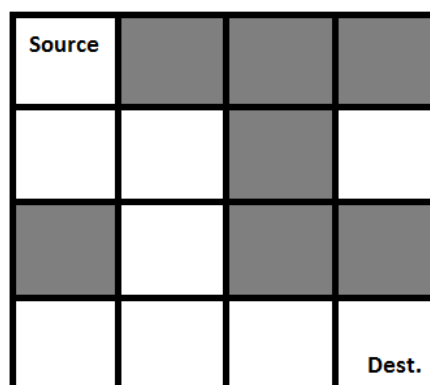


Gambar 1.3-2. Minimum Spanning Tree

Ada banyak kasus penggunaan spanning tree. Salah satu contohnya adalah perusahaan telekomunikasi yang mencoba memasang kabel di lingkungan baru. Jika dibatasi untuk mengubur kabel hanya di sepanjang jalur tertentu (misalnya jalan raya), maka akan ada grafik yang berisi titik-titik (misalnya rumah) yang dihubungkan oleh jalur tersebut. Beberapa jalur mungkin lebih mahal, karena lebih panjang, atau membutuhkan kabel untuk ditanam lebih dalam. Spanning tree untuk grafik tersebut akan menjadi bagian dari jalur yang tidak memiliki siklus tetapi masih menghubungkan setiap rumah sehingga terdapat jalur yang paling murah untuk memasang kabel.

1.4. Maze Problem

Maze atau labirin merupakan sebuah sistem jalur yang rumit, berliku-liku, serta memiliki banyak jalan buntu. Dari permasalahan tersebut dipilih jalur yang paling efektif tanpa menemui jalan buntu. Sebuah maze diibaratkan sebagai matriks biner $N * N$ dari blok dimana blok sumber adalah blok paling kiri atas yaitu, labirin $[0] [0]$ dan blok tujuan adalah blok paling kanan bawah yaitu, labirin $[N-1] [N-1]$. Dalam matriks labirin, 0 berarti blok tersebut adalah jalan buntu dan 1 berarti blok tersebut dapat digunakan di jalur dari sumber ke tujuan.



Gambar 1.4-1. Maze Problem

$$\begin{aligned} & \{1, 0, 0, 0\} \\ & \{1, 1, 0, 1\} \\ & \{0, 1, 0, 0\} \\ & \{1, 1, 1, 1\} \end{aligned}$$

A 4x4 grid illustrating a path from Source to Dest. The path is highlighted in green, starting at Source (top-left) and ending at Dest (bottom-right). The path moves right, then down, then right again. Obstacles are represented by gray squares.

Gambar 1.4-2. Solusi Maze Problem

Algoritme Dijkstra, (sesuai penemunya Edsger Dijkstra), adalah sebuah algoritma yang dipakai dalam memecahkan permasalahan jarak terpendek (shortest path problem) untuk sebuah graf berarah (directed graph).

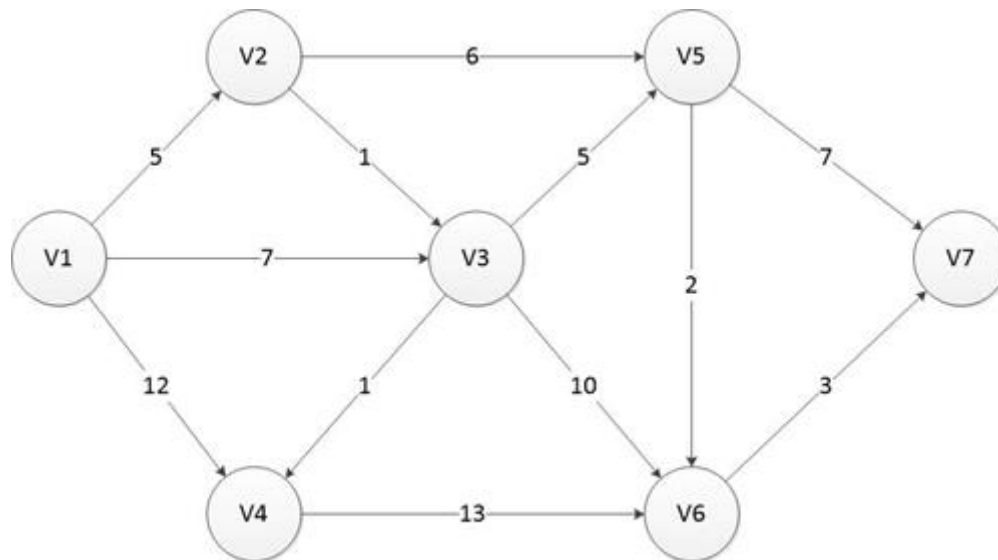
Permasalahan rute terpendek dari sebuah titik ke akhir titik lain adalah sebuah masalah klasik optimasi yang banyak digunakan untuk menguji sebuah algoritma yang diusulkan. Permasalahan rute terpendek dianggap cukup baik untuk mewakili masalah optimisasi, karena permasalahannya mudah dimengerti (hanya menjumlahkan seluruh edge yang dilalui) namun memiliki banyak pilihan solusi.

Menurut Andrew Goldberg peneliti Microsoft Research Silicon Valley, mengatakan ada banyak alasan mengapa peneliti terus mempelajari masalah pencarian jalan terpendek. “Jalan terpendek adalah masalah optimasi yang relevan untuk berbagai macam aplikasi, seperti jaringan routing, game, desain sirkuit, dan pemetaan”. Deskripsi matematis untuk grafik dapat diwakili $G = \{V, E\}$, yang berarti sebuah grafik (G) didefinisikan oleh satu set simpul (Vertex = V) dan koleksi Edge (E).

Algoritma Dijkstra bekerja dengan membuat jalur ke satu simpul optimal pada setiap langkah. Jadi pada langkah ke n, setidaknya ada n node yang sudah kita tahu jalur terpendek. Langkah-langkah algoritma Dijkstra dapat dilakukan dengan langkah-langkah berikut:

1. Tentukan titik mana yang akan menjadi node awal, lalu beri bobot jarak pada node pertama ke node terdekat satu per satu, Dijkstra akan melakukan pengembangan pencarian dari satu titik ke titik lain dan ke titik selanjutnya tahap demi tahap.
2. Beri nilai bobot (jarak) untuk setiap titik ke titik lainnya, lalu set nilai 0 pada node awal dan nilai tak hingga terhadap node lain (belum terisi) 2.
3. Set semua node yang belum dilalui dan set node awal sebagai “Node keberangkatan”
4. Dari node keberangkatan, pertimbangkan node tetangga yang belum dilalui dan hitung jaraknya dari titik keberangkatan. Jika jarak ini lebih kecil dari jarak sebelumnya (yang telah terekam sebelumnya) hapus data lama, simpan ulang data jarak dengan jarak yang baru
5. Saat kita selesai mempertimbangkan setiap jarak terhadap node tetangga, tandai node yang telah dilalui sebagai “Node dilewati”. Node yang dilewati tidak akan pernah di cek kembali, jarak yang disimpan adalah jarak terakhir dan yang paling minimal bobotnya.
6. Set “Node belum dilewati” dengan jarak terkecil (dari node keberangkatan) sebagai “Node Keberangkatan” selanjutnya dan ulangi langkah e.

Sebagai contoh hitunglah Jarak terdekat dari V1 ke V7 pada gambar berikut ini.



Gambar 1.5-1. Shortest Path Problem

Hasil setiap stepnya dapat dilihat pada tabel berikut ini.

Iteration	Unvisited (Q)	Visited (S)	Current	Node : Min = (dist[node], prev[node])iteration						
				V1	V2	V3	V4	V5	V6	V7
	Initialization {V1,V2,V3,V4,V5,V6,V7}	{-}		(0,-)0	(∞,-)0	(∞,-)0	(∞,-)0	(∞,-)0	(∞,-)0	(∞,-)0
1	{V2,V3,V4,V5,V6,V7}	{V1}	V1		(5,V1)1	(7,V1)1	(12,V1)1	(∞,V1)1	(∞,V1)1	(∞,V1)1
2	{V3,V4,V5,V6,V7}	{V1,V2}	V2			(6,V2)2	(12,V1)1	(11,V2)2	(∞,V2)2	(∞,V2)2
3	{V4,V5,V6,V7}	{V1,V2,V3}	V3				(7,V3)3	(11,V3)3	(16,V3)3	(∞,V3)3
4	{V5,V6,V7}	{V1,V2,V3,V4}	V4					(11,V3)3	(16,V3)3	(∞,V3)3
5	{V6,V7}	{V1,V2,V3,V4,V5}	V5						(13,V5)5	(18,V5)5
6	{V7}	{V1,V2,V3,V4,V5,V6}	V6							(16,V6)6

Tabel 1.5-1. Solusi Shortest Problem

Dengan demikian jarak terpendek dari V1 ke V7 adalah 16 dengan jalur V1->V2->V3->V5->V6->V7

1.6. Travelling Salesmen Problem

Travelling salesman problem (TSP) adalah suatu permasalahan mencari jarak terpendek ketika salesman akan mengunjungi beberapa kota tanpa mendatangi kota yang sama lebih dari

satu kali. Dan dia harus mulai dari dan kembali ke kota asal. Tujuan dari adanya ini selain untuk mencari rute terpendek yaitu untuk mendapat biaya yang paling sedikit. Solusi dari permasalahan STP ini akan sangat membantu perusahaan dalam mengirim barang, surat ataupun yang lainnya. Perusahaan akan lebih diuntungkan dari segi waktu dan dana. Konsep permasalahan TSP memiliki aturan sebagai berikut:

- a. Harus mengunjungi setiap kota tepat satu kali.
- b. Semua kota harus dikunjungi dalam satu kali perjalanan (tour).
- c. Dimulai dan diakhiri pada kota yang sama (kota awal).

Implementasi beberapa algoritma terhadap TSP antara lain:

1. Algoritma Greedy

Konsep dari Algoritma ini adalah mencari rute terpendek dari kota satu ke kota selanjutnya tanpa memperhatikan konsekuensi setelahnya. Sehingga algoritma greedy tidak bisa memberikan solusi yang optimal

2. Algoritma Artificial Bee Colony (ABC)

Pada algoritma ABC ini terinspirasi dari perilaku sekumpulan lebah madu yang mencari makanan. Namun pada algoritma ini hanya untuk kota yang terbatas.

3. Algoritma Cheapest Insertion Heuristics (CIH)

Algoritma ini merupakan kombinasi dari basis data yang mana basis data ini digunakan untuk proses penyimpanan data proses yang akan mempermudah pengambilan informasi

4. Algoritma Genetika

Algoritma ini yang menghasilkan hasil yang optimal karena dapat menghasilkan rute yang paling optimum. Konsep algoritma genetika sendiri adalah algoritma pencarian heuristik yang didasarkan pada mekanisme evolusi biologis. Keberagaman pada evolusi biologis adalah variasi dari kromosom dalam individu organisme.

```
import tsp
matrix=[[1,2,3,4],
        [5,6,7,8],
        [9,10,11,12],
        [13,14,15,16]]
r=range(len(matrix))

shortestpath={(i,j):matrix[i][j] for i in r for j in r}
print(tsp.tsp(r,shortestpath))

(34.0, [0, 3, 1, 2])
```

Gambar 1.6-1. Source Code TSP

BAB II

TOPIK PROJECT

2.1. Topik Project








Teknologi informasi saat ini sudah menjadi hal yang tidak dapat dipisahkan dari kehidupan manusia, bahkan saat ini teknologi sudah menjadi kebutuhan serta menjadi tuntutan dikarenakan perkembangan zaman. Penggunaan teknologi sudah mencakup hampir seluruh aspek kehidupan kita, baik itu secara individu maupun organisasi kini semuanya sangat membutuhkan teknologi. Pada laporan ini kami akan menekankan penerapan teknologi dalam suatu organisasi profit yang bergerak di bidang ekspedisi pengiriman barang menggunakan kargo.

Topik project kali ini, kami mengimplementasikan permasalahan fractional knapsack problem. Knapsack problem merupakan sebuah masalah yang berhubungan dengan persoalan optimasi pemilihan benda mana yang bisa ditampung ke dalam suatu wadah berkapasitas terbatas. Optimasi tersebut bertujuan untuk memudahkan memilih benda mana yang dapat memberikan keuntungan yang lebih besar jika dimasukkan kedalam suatu wadah berkapasitas terbatas. Sedangkan unbounded knapsack problem adalah penyelesaian permasalahan dimana semua benda yang menjadi permasalahan diasumsikan berjumlah tidak terbatas, sehingga berapapun jumlah benda yang menjadi permasalahan dianggap dapat ditampung ke dalam wadah selama masih memenuhi kapasitas wadah tersebut.

Optimasi pemilihan armada dengan bobot barang merupakan suatu factor yang penting dalam layanan jasa pengiriman/ekspedisi. Dalam menjalankan proses bisnisnya, masih banyak perusahaan penyedia jasa/layanan pengiriman seringkali mengalami beberapa kendala yang dapat merugikan perusahaan itu sendiri. Salah satu kendala yang dihadapi oleh perusahaan yaitu kurangnya kontrol dalam melakukan pemilihan barang yang akan dikirim (optimasi) dan armada yang akan mengangkutnya. Setiap menerima barang dari konsumen, staf perusahaan hanya mencatat data pengirim dan penerima dari barang yang akan dikirim tanpa memahami kapasitas dari kendaraan yang akan mengirim. Masalah terjadi saat melakukan muat barang, dimana barang akan sebanyak mungkin dimasukkan ke dalam kendaraan hingga terkadang melebihi kapasitas muatan kendaraan. Hal ini menyebabkan kerugian pada perusahaan karena bobot barang yang diangkut tidak sesuai dengan armada yang dipilih. Selain itu, pemilihan yang tidak optimal ini juga akan berdampak pada sisi user yang nantinya akan menerima biaya

pengiriman yang terlalu besar. Oleh karena itu, disini kami akan membuat sebuah aplikasi bernama Go Cargo untuk mengatasi permasalahan tersebut. Aplikasi ini nantinya dirancang dengan menerapkan algoritma knapsack problem dan berupa GUI (*Graphical User Interface*).

Tujuan dari pembuatan aplikasi ini untuk mempermudah proses bisnis pengiriman barang. Aplikasi ini akan memberikan pilihan armada yang tepat dalam membantu user untuk mengirimkan barang sesuai dengan bobot barang dan jarak pengiriman. Hal ini sesuai dengan definisi dari knapsack problem yang memilih barang berdasarkan keuntungan maksimal. Keuntungan maksimal ini nantinya diperoleh oleh user dan penyedia jasa. Dari sisi user, keuntungan didapatkan karena pemilihan armada sesuai dengan bobot barang yang dikirim sehingga biaya pengiriman tidak terlalu besar. Sedangkan, dari sisi penyedia jasa keuntungannya didapatkan dari memaksimalkan kapasitas atau daya tampung dari setiap armada sehingga bisa mengirimkan barang dengan jumlah banyak sekaligus dalam sekali pengiriman. Berikut daftar armada yang nantinya digunakan dalam pengiriman kargo:

Armada	Kapasitas	Ukuran
 Tronton Box	15.000 kg	L: 9.6 m W: 2.4 m H: 2.2 m
 Fuso Berat	8.000 kg	L: 5.7 m W: 2.3 m H: 2.2 m
 Fuso Ringan	5.000 kg	L: 5.0 m W: 2.1 m H: 2.0 m
 Engkel Box atau Engkel Bak	2.200 kg	L: 3.1 m W: 1.7 m H: 1.7 m
 Pickup	800 kg	L: 2.0 m W: 1.56 m H: 1.2 m
 Van	720 kg	L: 2.1 m W: 1.56 m H: 1.45 m
 Ekonomi	150 kg	L: 1.0 m W: 0.9 m H: 0.75 m

Gambar 2.1-1 Daftar Armada Go Cargo

Alur dari aplikasi yang dibuat nantinya yaitu user atau pengguna yang akan mengirimkan barang menginputkan jumlah barang yang akan dikirim dengan memasukkan detail tiap barang dari berat, panjang, lebar, dan tinggi barang. Kemudian, user disuruh menginputkan tempat tujuan pengiriman barang. Setelah itu, program akan memberikan pilihan armada yang sesuai

dengan user menggunakan knapsack problem. Terakhir, user akan memilih armada dari rekomendasi aplikasi dengan biaya yang telah tertera. Berikut sketch dari program aplikasi yang kelompok kami buat nantinya :



Gambar 2.1-2 Sketch Program Aplikasi

Harapan dari perancangan aplikasi ini untuk membantu meningkatkan kinerja dan kualitas pelayanan perusahaan sehingga dapat menguntungkan kedua belah pihak baik dari users maupun penyedia jasa.

2.2. Tujuan Pembuatan Aplikasi

Pembuatan serta pengembangan program ini memiliki tujuan antara lain adalah sebagai berikut :

1. Untuk meningkatkan kinerja dan layanan pengiriman barang pada perusahaan ekspedisi
2. Untuk mempermudah proses bisnis layanan pengiriman barang
3. Untuk mempermudah customer dalam mengirimkan barang
4. Untuk menyesuaikan pemilihan armada dengan bobot barang sesuai kapasitasnya

2.3. Batasan Aplikasi

Pada proses pembuatan serta pengembangannya, program ini memiliki beberapa batasan antara lain sebagai berikut :

1. Aplikasi ini terbatas hanya memiliki 7 armada pengiriman dengan masing-masing kapasitas maksimalnya sudah ditentukan.
2. Aplikasi ini hanya dapat mengirimkan barang dengan kapasitas maksimal 18.000 kg
3. Pengguna sebagai customer terbatas akan atau hanya dapat menggunakan program ini untuk keperluan pengiriman barang. Pengguna tidak dapat melakukan cancel barang apabila terjadi kesalahan pengiriman.
4. Proses pengiriman barang oleh armada hanya dapat dilakukan secara bergantian. Armada yang sedang melakukan proses pengiriman barang tidak dapat mengirimkan barang lain, sebelum barang yang sedang dikirimkan sudah sampai ditempat tujuan.

BAB III

RANCANGAN APLIKASI

3.1. Kebutuhan Sistem

Aplikasi yang kami rancang ini dimaksudkan untuk user memberikan pilihan armada yang tepat dalam mengirimkan barang sesuai dengan bobot barang dan jarak pengiriman. Untuk memenuhi tujuan tersebut, maka kami memberikan beberapa fitur (layanan) yang dapat dimanfaatkan oleh user, fitur-fitur yang kami rancang adalah sebagai berikut :

User sebagai customer

- a. Aplikasi ini memiliki fitur login untuk melakukan proses identifikasi pengguna.
- b. Aplikasi ini memiliki fitur registrasi akun untuk membuat akun baru yang ditujukan kepada pengguna baru yang belum memiliki akun untuk login.
- c. Aplikasi ini memiliki fitur untuk melakukan pengiriman barang sesuai dengan menginputkan data-data pengiriman yang diperlukan.
- d. Aplikasi ini memiliki fitur untuk memeriksa pengiriman (cek pengiriman).
- e. Aplikasi ini memiliki fitur untuk menampilkan profile dari customers.

User sebagai admin atau pengirim

- a. Aplikasi ini memiliki fitur login untuk melakukan proses identifikasi pengguna.
- b. Aplikasi ini memiliki fitur untuk memproses pengiriman dari customer dengan memilih armada sesuai kapasitas dan bobot barang yang akan dikirimkan
- c. Aplikasi ini memiliki fitur untuk melihat barang apa saja yang perlu untuk dikirimkan beserta detail pengirimannya.
- d. Aplikasi ini memiliki fitur menampilkan profile dari admin pengirim.
- e. Aplikasi ini memiliki fitur untuk menambahkan akun baru untuk pengirim baru.

3.2. Struktur Data

Aplikasi ini menggunakan beberapa jenis data dan tipe data yang berbeda-beda dalam pengoperasiannya. Untuk lebih jelasnya dapat melihat tabel dibawah ini.

No.	Fitur	Data	Tipe Data
1	Login	Email Password	String String

2	Registrasi Akun	Nama Email Password Jenis Kelamin Umur	String String String String Integer
3	Tambah Data Pengiriman Barang	Kota Asal Kota Tujuan Berat Barang (kg)	String String Integer
4	Lihat Data Pengiriman Barang	Nama Pengirim Kota Asal Kota Tujuan Berat Barang (kg) Status Pengiriman	String String String Integer String
5	Lihat Data Profil Pengguna	Nama Email Jenis Kelamin Umur	String String String Integer
6	Kirim Barang	Kota Asal Tipe Kendaraan Kapasitas Kendaraan Status Kendaraan	String String String String
7	Lihat Data Barang yang Belum Ter kirim	Nama Pelanggan Kota Asal Kota Tujuan Berat Barang (kg)	String String String Integer
8	Lihat Data Profil Pengirim	Nama Email Jenis Kelamin Umur	String String String Integer
9	Tambah Data Pengirim	Nama Email Password Jenis Kelamin	String String String String

		Umur	Integer
--	--	------	---------

3.3. Rancangan Langkah Penggunaan Aplikasi

Aplikasi yang kami buat berupa *Graphical User Interface* (GUI). Pengoperasiannya cukup dengan login terlebih dahulu agar aplikasi dapat melakukan proses identifikasi user untuk mendapatkan hak akses. Proses ini juga berguna untuk menjaga keamanan data pengguna. Setelah proses login pengguna akan diberikan pilihan menu untuk pengoperasian fitur utama aplikasi. Pilihan menu yang diberikan dapat berbeda sesuai peran dari user(customer atau admin pengirim).

Berikut alur pengoperasian aplikasi untuk **customer**, dimulai dari halaman login aplikasi.

1. Registrasi Akun

User dapat melakukan registrasi akun terlebih dahulu apabila belum mempunyai akun untuk login ke aplikasi. Fitur ini akan meminta user untuk mengisi username dan password sesuai keinginan dari user. Fitur ini tidak diwajibkan atau user dapat melewati langkah ini apabila sudah memiliki akun untuk login ke aplikasi.

2. Login

User akan diminta memasukkan username dan password yang sudah terdaftar di aplikasi untuk mendapatkan hak akses masuk ke aplikasi.

Setelah berhasil login kedalam aplikasi, users dapat mengakses berbagai fitur utama yang tersedia pada aplikasi.

3. Menu pengiriman barang (Beranda)

Fitur ini dapat diakses pada menu home. Pada fitur ini user (customers) dapat melakukan pengiriman barang dengan menginputkan detail pengiriman yang dibutuhkan.

4. Cek Pengiriman

Fitur ini dapat diakses apabila user memilih menu cek pengiriman. Pada fitur ini user dapat melakukan cheking terhadap barangnya. User dapat melihat status dari barangnya, apakah barang yang telah dikirimkan ke ekspedisi sudah diproses atau belum.

5. Profile

Fitur ini dapat diakses apabila user memilih menu profile. Fitur ini menampilkan profile dari user itu sendiri.

6. Keluar

Fitur ini berfungsi untuk menutup menu aplikasi dan secara otomatis akan mengembalikan user menuju ke halaman awal login. User dapat melakukan close aplikasi untuk menghentikan program.

Berikut alur pengoperasian aplikasi untuk **admin sebagai pengirim**, yang dimulai dari halaman login aplikasi.

1. Login

User akan diminta memasukkan username dan password yang sudah terdaftar di aplikasi untuk mendapatkan hak akses masuk ke aplikasi.

Masuk ke aplikasi untuk mengakses fitur utama.

2. Menu pengiriman barang (Beranda)

Fitur ini dapat diakses oleh user dengan memilih menu home. Pada fitur ini user dapat mengirimkan barang sesuai bobot barang (yang dapat dilihat pada menu storage) dan kapasitas armadanya.

3. Storage (detail barang)

Fitur ini dapat diakses apabila user memilih menu storage. Pada fitur ini user dapat melakukan check barang apa saja yang sekiranya perlu untuk dikirimkan.

4. Profile dan Pendaftaran akun baru untuk pengirim

Fitur ini dapat diakses apabila user memilih menu profile. Fitur ini menampilkan profile dari admin sebagai pengirim. Selain itu di menu ini juga terdapat fitur untuk menambahkan akun baru pengirim.

5. Keluar

Fitur ini berfungsi untuk menutup menu aplikasi dan secara otomatis akan mengembalikan user menuju ke halaman awal login. User dapat melakukan close aplikasi untuk menghentikan program.

BAB IV

IMPLEMENTASI RANCANGAN

Aplikasi Go-Cargo dibuat menggunakan bahasa pemrograman python dengan menggunakan module tkinter untuk membuat *Graphical User Interface* pada aplikasi. Data pada aplikasi ini disimpan oleh database mongodb yang dapat diakses melalui API. Adapun kode program dari aplikasi Go-Cargo akan kami jelaskan dalam bentuk sub-bab berdasarkan fitur pada aplikasi.

4.1. Import Module

Kode Program go-cargo.py

```
import tkinter as tk
from tkinter import messagebox, StringVar
import requests
from PIL import Image, ImageTk
from io import BytesIO
from datetime import datetime

url = "https://go-cargo.herokuapp.com"
```

Pada baris awal, kami menggunakan beberapa module antarlain tkinter, requests, image, io dan datetime. Module tkinter digunakan untuk membuat *Graphical User Interface* pada aplikasi. Module request digunakan untuk mengirimkan HTTP requests. Module image digunakan untuk menampilkan gambar pada aplikasi, Module io digunakan untuk memanipulasi string dan byte data dalam memori. Sedangkan, module datetime digunakan untuk mendapatkan waktu sekarang.

4.2. Autentikasi

Kode Program go-cargo.py

```
class Auth(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        self.resizable(False, False)
        self.title("Go Cargo")
        self.geometry("1305x780")

        container = tk.Frame(self)
        container.pack(side="left", fill="both", expand=True)
```

```

        container.rowconfigure(0, weight=1)
        container.columnconfigure(0, weight=1)

        self.frames = {}

        self.frames["Login"] = Login(parent=container, controller=self)
        self.frames["Register"] = Register(parent=container, controller=self)
    )

    self.frames["Login"].grid(row=0, column=0, sticky="nsew")
    self.frames["Register"].grid(row=0, column=0, sticky="nsew")

    self.show_frame("Login")

def show_frame(self, page_name):
    frame = self.frames[page_name]
    frame.tkraise()

def backgroundImage(self):
    urlImage = requests.get(
        "https://firebasestorage.googleapis.com/v0/b/go-cargo-12c47.appspot.com/o/login.jpg?alt=media&token=2b0be545-df36-4e4f-bda1-86bd6427a5ea")
    img_data = urlImage.content
    img_resize = Image.open(BytesIO(img_data)).resize(
        (1305, 780), Image.ANTIALIAS)
    render = ImageTk.PhotoImage(img_resize)
    img = tk.Label(self, image=render)
    img.image = render
    img.place(x=0, y=0)

def handleLogin(self):
    email = self.entry_email.get()
    password = self.entry_password.get()

    response = requests.get(f"{url}/login/{email}/{password}")

    if response.status_code == 200:
        messagebox.showinfo("Informasi", "Login berhasil")

        user = response.json()

        global userName
        global userEmail

        userName = user["name"]
        userEmail = user["email"]

```



```

        self.controller.destroy()

        if user["role"] == "customer":
            UserMenu()
        elif user["role"] == "shipper":
            AdminMenu()

    else:
        messagebox.showerror("Informasi", "Login gagal")

def handleRegister(self, role):
    name = self.entry_name.get()
    email = self.entry_email.get()
    password = self.entry_password.get()
    gender = self.var_gender.get()
    age = int(self.entry_age.get())

    response1 = requests.get(f"{url}/users/{email}")

    if response1.status_code == 200:
        messagebox.showerror("Informasi", "Email sudah digunakan")
    else:
        userData = {
            "name": name,
            "email": email,
            "password": password,
            "gender": gender,
            "age": age,
            "role": role,
        }

        response2 = requests.post(f"{url}/users", userData)

        if response2.status_code == 200:
            messagebox.showinfo("Informasi", "Akun berhasil didaftarkan")
        else:
            messagebox.showerror("Informasi", "Akun gagal didaftarkan")

    self.entry_name.delete(0, "end")
    self.entry_email.delete(0, "end")
    self.entry_password.delete(0, "end")
    self.entry_age.delete(0, "end")

```

Authentication digunakan untuk menangani login dan register pada aplikasi. Pada bagian register user disuruh menginputkan nama, email, password, jenis kelamin dan umur. Email yang didaftarkan harus belum terdaftar pada aplikasi. Sedangkan pada bagian login, user disuruh menginputkan email dan password yang sudah terdaftar pada aplikasi. Jika berhasil maka akan dialihkan ke dalam menu aplikasi sesuai pada role akun tersebut.

4.3. Fitur Login

Kode Program go-cargo.py

```
class Login(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        Auth.backgroundImg(self)

        tk.Label(self, text="Email", width=20,
                  font=("bold", 10)).place(x=150, y=350)
        self.entry_email = tk.Entry(self)
        self.entry_email.place(x=280, y=350)

        tk.Label(self, text="Password", width=20,
                  font=("bold", 10)).place(x=150, y=400)
        self.entry_password = tk.Entry(self, show="*")
        self.entry_password.place(x=280, y=400)

        tk.Button(
            self, text="Login", bg="dodger blue", fg="white", command=lambda
: Auth.handleLogin(self)
        ).place(x=250, y=450, height=50, width=100)
        tk.Button(
            self, text="Register", command=lambda: controller.show_frame("Re
gister")
        ).place(x=320, y=520, height=50, width=100)
        tk.Button(
            self,
            text="Exit",
            bg="brown",
            fg="white",
            command=lambda: controller.destroy(),
        ).place(x=200, y=520, height=50, width=100)
```

Kode diatas digunakan untuk membuat Graphical User Interface pada menu login dengan menggunakan tkinter.

4.4. Fitur Registrasi

Kode Program go-cargo.py

```
class Register(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.controller = controller

        Auth.backgroundImg(self)

        tk.Label(self, text="Daftar Akun Baru",
                  width=20, font=("bold", 20)).place(x=150, y=280)

        tk.Label(self, text="Nama Lengkap",
                  width=20, font=("bold", 10)).place(x=150, y=340)
        self.entry_name = tk.Entry(self)
        self.entry_name.place(x=300, y=340)

        tk.Label(self, text="Email", width=20,
                  font=("bold", 10)).place(x=150, y=370)
        self.entry_email = tk.Entry(self)
        self.entry_email.place(x=300, y=370)

        tk.Label(self, text="Password", width=20,
                  font=("bold", 10)).place(x=150, y=400)
        self.entry_password = tk.Entry(self, show="*")
        self.entry_password.place(x=300, y=400)

        tk.Label(self, text="Jenis Kelamin",
                  width=20, font=("bold", 10)).place(x=150, y=430)

        self.var_gender = StringVar()

        self.select_gender = tk.Radiobutton(
            self, text="Pria", padx=5, value="pria", variable=self.var_gende
            r
        ).place(x=280, y=430)
        self.select_gender = tk.Radiobutton(
            self, text="Wanita", padx=20, value="wanita", variable=self.var_
            gender
        ).place(x=330, y=430)

        tk.Label(self, text="Umur:", width=20,
                  font=("bold", 10)).place(x=150, y=460)

        self.entry_age = tk.Entry(self)
        self.entry_age.place(x=300, y=460)
```

```

tk.Button(
    self,
    text="Daftar",
    width=20,
    bg="dodger blue",
    fg="white",
    command=lambda: Auth.handleRegister(self, "customer"),
).place(x=150, y=520)

tk.Button(
    self,
    text="Back to login",
    width=20,
    command=lambda: controller.show_frame("Login"),
).place(x=320, y=520)

```

Kode diatas digunakan untuk membuat Graphical User Interface pada menu register dengan menggunakan tkinter.

4.5. Sidebar

Kode Program go-cargo.py

```

ScrollOnItemsList = []

class ScrollBar(tk.Frame):
    def __init__(self, parent):
        self.height = 780
        self.width = 200

        tk.Frame.__init__(self, parent, width=self.width,
                           height=self.height, bg="grey")
        self.place(x=0, y=-2)

        self.last_delta = 0
        parent.update()

        self.c_frame = tk.Frame(self, width=self.width,
                                  height=self.height, bg="grey")
        self.c_frame.place(x=0, y=0)
        self.c_Canvas = tk.Canvas(
            self.c_frame, width=self.width, height=self.height, bd=-
            2, bg="#232323"
        )

```

```

        scrollbar = tk.Scrollbar(
            self.c_frame, orient="vertical", command=self.c_Canvas.yview, bg
="grey"
        )
        self.scrollframe = tk.Frame(self.c_Canvas)
        self.scrollframe.place(x=0, y=0)

        self.scrollframe.bind(
            "<Configure>",
            lambda e: self.c_Canvas.configure(
                scrollregion=self.c_Canvas.bbox("all")),
        )
        self.c_Canvas.create_window(
            (0, 0), window=self.scrollframe, anchor="nw")

        self.c_Canvas.configure(yscrollcommand=scrollbar.set)
        self.c_Canvas.pack(side="left", fill="both", expand=True)
        scrollbar.pack(side="right", fill="y")

side_bar_tab_list = []

class SideBar(ScrollBar):
    def __init__(self, parent, *args, **kwargs):
        ScrollBar.__init__(self, parent)
        self.color = "#232323"

    def finish(self):
        self.select_first_tab()

    def add_button(self, text, command, tab=True):
        SideBarButton(self.scrollframe, text, command, tab=tab)

    def select_first_tab(self):
        i = side_bar_tab_list[0]
        i.click()

class SideBarButton(tk.Canvas):
    def __init__(self, parent, text, command, tab=True, *args, **kwargs):

        self.frame_color = "#232323"
        self.hover_color = "#4D4c4c"
        self.hover_border_color = "grey"
        self.is_tab = tab

        self.selected = False

```

```

self.command = command

tk.Canvas.__init__(
    self,
    parent,
    width=198,
    height=35,
    bg=self.frame_color,
    highlightthickness=1,
    highlightbackground=self.frame_color,
    *args,
    **kwargs,
)
self.pack()

self.text = tk.Label(
    self, text=text, font="Segoe 10", bg=self.frame_color, fg="light
grey"
)
self.text.place(x=40, y=10)

self.bind("<Enter>", self.hover)
self.bind("<Button-1>", self.click)
if self.is_tab == False:
    self.bind("<ButtonRelease-1>", self.unclick)

self.text.bind("<Enter>", self.hover)
self.text.bind("<Button-1>", self.click)
if self.is_tab == False:
    self.text.bind("<ButtonRelease-1>", self.unclick)
if self.is_tab:
    side_bar_tab_list.append(self)
ScrollOnItemsList.append(self)

def hover(self, event=None):
    if self.selected == False:
        self.bind("<Leave>", self.unhover)
        self.config(
            highlightbackground=self.hover_border_color, bg=self.hover_c
olor
        )
        self.text.config(bg=self.hover_color)

def unhover(self, event=None):
    self.config(highlightbackground=self.frame_color, bg=self.frame_colo
r)
    self.text.config(bg=self.frame_color)

```

```

def click(self, event=None):

    if self.is_tab:
        self.bind("<Leave>", str)
        for i in side_bar_tab_list:
            i.unhover()
            i.selected = False

        self.selected = True

        self.config(bg=self.hover_border_color)
        self.text.config(bg=self.hover_border_color)

        self.command()

def unclick(self, event=None):
    self.selected = False
    self.config(bg=self.hover_color)
    self.text.config(bg=self.hover_color)

active_page = []

class Page(tk.Frame):
    def __init__(self, parent):
        self.width = 1350
        self.height = 780
        # remove old page
        try:
            active_page[0].delete()
        except:
            pass

        # create new tab
        tk.Frame.__init__(
            self, parent, bg="white", height=self.height, width=self.width
        )

        # place page
        self.place(x=0, y=0)

        active_page.append(self)

    def delete(self):
        global active_page
        active_page.remove(self)
        self.destroy()

```

Fitur sidebar digunakan agar pengguna dapat berpindah ke menu lainnya sesuai dengan role akun. Terdapat 2 role pada aplikasi ini yaitu customer (pelanggan) dan shipper (pengirim). Setiap role memiliki menu yang berbeda.

4.6. Menu Pelanggan

Kode Program go-cargo.py

```
class UserMenu(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        self.title("Customer Menu")
        self.resizable(False, False)
        self.geometry("1305x780")

        main_frame = tk.Frame(self, bg="white", width=1350, height=1000)
        main_frame.place(x=200, y=0)

        sidebar = SideBar(self)

        sidebar.add_button("Home", lambda: UserHome(main_frame))
        sidebar.add_button("Cek Pengiriman", lambda: UserShipping(main_frame))
        sidebar.add_button("Profile", lambda: UserProfile(main_frame))
        sidebar.add_button("Exit", lambda: exit(self))
        sidebar.finish()

    def exit(self):
        self.destroy()
```

Bagian menu pelanggan merupakan detail menu yang nantinya dimasukkan ke dalam sidebar. Terdapat 4 menu antara lain home, cek pengiriman, profil, dan exit.

4.6.1. Fitur Home

Kode Program go-cargo.py

```
class UserHome(Page):
    def __init__(self, parent):

        # init page/ delete old page
        Page.__init__(self, parent)

        response = requests.get(f"{url}/port")
```



```

self.results = response.json()

port = []

for index, element in enumerate(self.results):
    port.append(element["city_name"])

self.optOrigin = tk.StringVar(self) # variable
self.optOrigin.set(port[0]) # default value

tk.Label(self, text="Pilih kota asal").place(x=20, y=30)
tk.OptionMenu(self, self.optOrigin, *port).place(x=20, y=60)

self.optDestination = tk.StringVar(self) # variable
self.optDestination.set(port[0]) # default value

tk.Label(self, text="Pilih kota tujuan").place(x=180, y=30)
tk.OptionMenu(self, self.optDestination,
               *port).place(x=180, y=60)

tk.Label(self, text="Berat (kg)").place(x=330, y=30)
self.entry_weight = tk.Entry(self)
self.entry_weight.place(x=330, y=60)

tk.Button(
    self,
    text="submit",
    bg="dodger blue",
    fg="white",
    command=self.handleShipping,
).place(x=500, y=60, height=30, width=100)

def handleShipping(self):
    origin = self.optOrigin.get()
    destination = self.optDestination.get()
    weight = int(self.entry_weight.get())

    shippingData = {
        "customer": userName,
        "origin": origin,
        "destination": destination,
        "weight": weight,
    }

    response = requests.post(f"{url}/shipping", shippingData)
    shipping = response.json()

```

```

if response.status_code == 200:
    messagebox.showinfo("Informasi", "Barang berhasil ditambahkan")
    self.entry_weight.delete(0, "end")

    tk.Label(self, text="Detail Pengiriman :").place(x=20, y=120)

    tk.Label(self, text="Kota Asal").place(x=20, y=150)
    tk.Label(
        self, text=shipping["origin"], anchor="w", width=100, bg="white"
    ).place(x=150, y=150)

    tk.Label(self, text="Kota Tujuan").place(x=20, y=180)
    tk.Label(
        self, text=shipping["destination"], anchor="w", width=100, bg="white"
    ).place(x=150, y=180)

    tk.Label(self, text="Berat Barang").place(x=20, y=210)
    tk.Label(
        self, text=shipping["weight"], anchor="w", width=100, bg="white"
    ).place(x=150, y=210)

    tk.Label(self, text="Biaya Pengiriman").place(x=20, y=240)
    tk.Label(
        self, text=shipping["cost"], anchor="w", width=100, bg="white"
    ).place(x=150, y=240)

    tk.Label(self, text="Estimasi Pengiriman").place(x=20, y=270)

    tk.Label(
        self,
        text=f"{shipping['estimation']} hari",
        anchor="w",
        width=100,
        bg="white",
    ).place(x=150, y=270)
else:
    messagebox.showerror("Informasi", "Barang gagal ditambahkan")

```

Pada bagian home, pelanggan dapat menambahkan barang untuk dikirim. Pelanggan nantinya disuruh mengisi kota asal, kota tujuan dan berat barang. Jika sudah terisi semua bisa langsung

klik tombol submit. Setelah itu, informasi terkait barang yang dikirim dengan biaya dan waktu pengiriman akan muncul.

4.6.2. Fitur Lihat Pengiriman

Kode Program go-cargo.py

```
class UserShipping(Page):
    def __init__(self, parent):
        # init page/ delete old page
        Page.__init__(self, parent)

        tk.Label(
            self, width=20, text="pengirim", anchor="w", bg="white").place(x=20, y=60)
        tk.Label(
            self, width=20, text="kota asal", anchor="w", bg="white").place(x=180, y=60)
        tk.Label(
            self, width=20, text="kota tujuan", anchor="w", bg="white"
        ).place(x=320, y=60)
        tk.Label(
            self, width=20, text="berat (kg)", anchor="w", bg="white"
        ).place(x=480, y=60)
        tk.Label(
            self, width=20, text="status", anchor="w", bg="white").place(x=580, y=60)

        response = requests.get(f"{url}/shippingcust/{userName}")
        results = response.json()

        for i in range(len(results)):
            self.labelShip = tk.Label(
                self, width=20, text=results[i]["shipper"], anchor="w", bg="white"
            )
            self.labelShip.place(x=20, y=(i + 1) * 30 + 60)

            if results[i]["shipper"] == "":
                self.labelShip.config(text="-")

            tk.Label(
                self, width=20, text=results[i]["origin"], anchor="w", bg="white"
            ).place(x=180, y=(i + 1) * 30 + 60)
            tk.Label(
                self, width=20, text=results[i]["destination"], anchor="w",
                bg="white"
            ).place(x=320, y=(i + 1) * 30 + 60)
```

```

        tk.Label(
            self, width=20, text=results[i]["weight"], anchor="w", bg="white"
        ).place(x=480, y=(i + 1) * 30 + 60)

        self.labelStatus = tk.Label(self, width=20, anchor="w", bg="white")
        self.labelStatus.place(
            x=580, y=(i + 1) * 30 + 60)
        if results[i]["status"] == 1:
            self.labelStatus.config(text="Sudah terkirim")
        else:
            self.labelStatus.config(text="Proses pengiriman")

```

Fitur lihat pengiriman digunakan untuk pelanggan agar bisa melihat nama pengirim, detail pengiriman dan status pengiriman setiap barang yang dimasukkan pelanggan. Jika berhasil terkirim maka akan muncul nama pengirim dan status pengiriman berubah menjadi sudah terkirim. Sedangkan, jika belum terkirim maka nama pengirim akan kosong dan status pengiriman masih dalam proses.

4.6.3. Fitur Lihat Profil

Kode Program go-cargo.py

```

class UserProfile(Page):
    def __init__(self, parent):
        # init page/ delete old page
        Page.__init__(self, parent)

        response = requests.get(f"{url}/users/{userEmail}")
        results = response.json()

        tk.Label(
            self,
            text="Detail Profile",
            width=20,
            bg="white",
            anchor="w",
            font=("bold", 20),
        ).place(x=20, y=60)

        tk.Label(self, text="Nama", width=20,
            bg="white", anchor="w").place(x=20, y=120)
        tk.Label(

```

```

        self, text=results["name"], width=20, bg="white", anchor="w").place(x=100, y=120)

        tk.Label(self, text="Email",
                  width=20, bg="white", anchor="w").place(x=20, y=150)
        tk.Label(
            self, text=results["email"], width=20, bg="white", anchor="w").place(x=100, y=150)

        tk.Label(
            self, text="Jenis Kelamin", width=20, bg="white", anchor="w"
        ).place(x=20, y=180)
        tk.Label(
            self, text=results["gender"], width=20, bg="white", anchor="w"
        ).place(x=100, y=180)

        tk.Label(self, text="Umur", width=20,
                  bg="white", anchor="w").place(x=20, y=210)

        tk.Label(
            self, text=results["age"], width=20, bg="white", anchor="w").place(x=100, y=210)

```

Pada fitur lihat profile, pelanggan dapat melihat detail profilnya antara lain nama, email, jenis kelamin dan umur. Disini pelanggan hanya bisa melihat profilnya saja dan tidak bisa merubah profilnya.

4.7. Menu Pengirim

Kode Program go-cargo.py

```

class AdminMenu(tk.Tk):
    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)

        self.title("Shipper Menu")
        self.resizable(False, False)
        self.geometry("1305x780")

        main_frame = tk.Frame(self, bg="white", width=1350, height=1000)
        main_frame.place(x=200, y=0)

        sidebar = SideBar(self)

        sidebar.add_button("Home", lambda: AdminHome(main_frame))
        sidebar.add_button("Storage", lambda: AdminStorage(main_frame))

```

```

        sidebar.add_button("Profile", lambda: AdminProfile(main_frame))
        sidebar.add_button("Exit", lambda: exit(self))
        sidebar.finish()

def exit(self):
    self.destroy()

```

Bagian menu pengirim merupakan detail menu yang nantinya dimasukkan ke dalam sidebar. Terdapat 4 menu antara lain home, storage (penyimpanan barang), profil, dan exit.

4.7.1. Fitur Home

Kode Program go-cargo.py

```

class AdminHome(Page):
    def __init__(self, parent):
        # init page/ delete old page
        Page.__init__(self, parent)

        response = requests.get(f"{url}/port")
        self.results = response.json()
        portData = []

        for index, element in enumerate(self.results):
            portData.append(element["city_name"])

        self.optPort = tk.StringVar(self) # variable
        self.optPort.set(portData[0]) # default value

        tk.Label(self, text="Pilih port").place(x=20, y=60)
        tk.OptionMenu(self, self.optPort, *portData).place(x=150, y=60)

        tk.Button(
            self, text="check", bg="dodger blue", fg="white", command=self.h
            andlePort
        ).place(x=350, y=60, height=30, width=100)

    def handlePort(self):
        tk.Label(
            self, width=20, text="type", anchor="w", bg="white").place(x=20,
            y=120)
        tk.Label(
            self, width=20, text="capacity (kg)", anchor="w", bg="white"
        ).place(x=180, y=120)
        tk.Label(
            self, width=20, text="status", anchor="w", bg="white").place(x=3
            20, y=120)

```

```

        tk.Label(
            self, width=20, text="action", anchor="w", bg="white").place(x=4
20, y=120)

        self.port = self.optPort.get()
        response = requests.get(f"{url}/port/{self.port}")
        results = response.json()

        for i in range(7):
            tk.Label(
                self,
                width=20,
                text=results["transport"][i]["type"],
                anchor="w",
                bg="white",
            ).place(x=20, y=(i + 1) * 30 + 120)

            tk.Label(
                self,
                width=20,
                text=results["transport"][i]["capacity"],
                anchor="w",
                bg="white",
            ).place(x=180, y=(i + 1) * 30 + 120)

            self.labelStatus = tk.Label(self, width=20, anchor="w", bg="whit
e")
            self.labelStatus.place(
                x=320, y=(i + 1) * 30 + 120)

            if results["transport"][i]["status"] == 0:
                self.labelStatus.configure(text="Ready")

            tk.Button(
                self,
                text="kirim",
                width=20,
                bg="dodger blue",
                command=lambda type=results["transport"][i][
                    "type"
                ], capacity=results["transport"][i]["capacity"]: self.ha
ndleSend(
                    type, capacity
                ),
            ).place(x=420, y=(i + 1) * 30 + 120)
        else:
            self.labelStatus.configure(text="On going")
            tk.Button(

```

```

        self,
        text="selesai",
        width=20,
        bg="yellow",
        command=lambda type=results["transport"][i][
            "type"
        ]: self.handleDone(type),
    ).place(x=420, y=(i + 1) * 30 + 120)

def handleSend(self, type, capacity):
    value = []
    weight = []
    item_id = []

    response1 = requests.get(f"{url}/storage/{self.port}/0")
    results = response1.json()

    for i in range(len(results)):
        deliveryDate = datetime.strptime(
            results[i]["deliveryDate"], "%Y-%m-%d %H:%M:%S"
        )
        nowDate = datetime.now()

        delivDate = int(deliveryDate.strftime("%Y%m%d%H%M%S"))
        nowInt = int(nowDate.strftime("%Y%m%d%H%M%S"))
        remaining = nowInt - delivDate

        value.append(remaining)
        weight.append(results[i]["weight"])
        item_id.append(results[i]["_id"])

    n = len(value)

    hasil = self.knapsack_brute(capacity, weight, value, n)

    if len(hasil[1]) == 0:
        messagebox.showerror("Informasi", "Tidak ada barang yang dikirim")
    else:
        tk.Label(
            self, width=20, text="customer", anchor="w", bg="white"
        ).grid(row=0, column=0)

        tk.Label(
            self, width=20, text="kota asal", anchor="w", bg="white"
        ).grid(row=0, column=1)

        tk.Label(

```



```

        self, width=20, text="kota tujuan", anchor="w", bg="white"
    ).grid(row=0, column=2)

    tk.Label(
        self, width=20, text="berat (kg)", anchor="w", bg="white"
    ).grid(row=0, column=3)

    listGoods = [[] for i in range(len(hasil[1]))]

    for i, v in enumerate(hasil[1]):
        requests.put(
            f"{url}/shipping/{item_id[v]}",
            {"shipper": userName},
        )

        listGoods[i].append(results[v]["customer"])
        listGoods[i].append(results[v]["origin"])
        listGoods[i].append(results[v]["destination"])
        listGoods[i].append(results[v]["weight"])

        for j in range(4):
            tk.Label(
                self, width=20, text=listGoods[i][j], anchor="w", bg
="white"

            ).grid(row=i + 1, column=j)
            tk.Label(
                self, width=20, text=listGoods[i][j], anchor="w", bg
="white"

            ).grid(row=i + 1, column=j)
            tk.Label(
                self, width=20, text=listGoods[i][j], anchor="w", bg
="white"

            ).grid(row=i + 1, column=j)
            tk.Label(
                self, width=20, text=listGoods[i][j], anchor="w", bg
="white"

            ).grid(row=i + 1, column=j)

    portData = {"type": type, "status": 1}

    response2 = requests.put(
        f"{url}/port/{self.port}", portData
    )

    if response2.status_code == 200:
        messagebox.showinfo("Informasi", "Barang telah dikirim")
    else:
        messagebox.showerror("Informasi", "Barang gagal dikirim")

```

```

def knapsack_brute(self, W, wt, val, n):
    if n == 0 or W == 0:
        return [0, []]

    elif wt[n - 1] > W:
        return self.knapsack_brute(W, wt, val, n - 1)

    simpan1 = self.knapsack_brute(W - wt[n - 1], wt, val, n - 1)
    simpan2 = self.knapsack_brute(W, wt, val, n - 1)

    nilai1 = val[n - 1] + simpan1[0]
    nilai2 = simpan2[0]

    nilai_max = max(nilai1, nilai2)

    if nilai_max == nilai1:
        simpan1[0] = nilai_max
        index_item = [n - 1]
        simpan1[1].extend(index_item)
        return simpan1
    else:
        return simpan2

def handleDone(self, type):
    portData = {"type": type, "status": 0}

    response2 = requests.put(f"{url}/port/{self.port}", portData)

    if response2.status_code == 200:
        messagebox.showinfo("Informasi", "Barang selesai dikirim")
    else:
        messagebox.showerror("Informasi", "Barang gagal dikirim")

```

Pada bagian home, pengirim dapat mengirimkan barang yang telah dimasukkan oleh pelanggan. Disini pengirim akan memilih port awal terlebih dahulu. Jika sudah klik check untuk melihat kendaraan yang siap pada port tersebut. Detail kendaraan dapat dilihat pada gambar 2.1. Setelah memilih kendaraan pengirim bisa klik tombol kirim. Selanjutnya program akan memilih barang sesuai dengan kapasitas kendaraan menggunakan algoritma brute force. Jika berhasil maka akan muncul detail barang yang dikirim.

4.7.2. Fitur Lihat Penyimpanan Barang

Kode Program go-cargo.py

```

class AdminStorage(Page):
    def __init__(self, parent):
        # init page/ delete old page
        Page.__init__(self, parent)

        response = requests.get(f"{url}/port")
        self.results = response.json()
        portData = []

        for index, element in enumerate(self.results):
            portData.append(element["city_name"])

        self.optPort = tk.StringVar(self) # variable
        self.optPort.set(portData[0]) # default value

        tk.Label(self, text="Pilih port").place(x=20, y=60)
        tk.OptionMenu(self, self.optPort, *portData).place(x=150, y=60)

        tk.Button(
            self, text="check", bg="dodger blue", fg="white", command=self.h
            andleStorage
        ).place(x=350, y=60, height=30, width=100)

    def handleStorage(self):
        tk.Label(
            self, width=20, text="customer", anchor="w", bg="white").place(x
            =20, y=120)
        tk.Label(
            self, width=20, text="kota asal", anchor="w", bg="white").place(
            x=180, y=120)
        tk.Label(
            self, width=20, text="kota tujuan", anchor="w", bg="white"
        ).place(x=320, y=120)
        tk.Label(
            self, width=20, text="berat (kg)", anchor="w", bg="white"
        ).place(x=450, y=120)

        origin = self.optPort.get()

        response = requests.get(f"{url}/storage/{origin}/0")
        results = response.json()

        if len(results) == 0:
            messagebox.showerror("informasi", "Tidak ada barang")
        else:
            for i in range(len(results)):
                tk.Label(

```

```

        self, width=20, text=results[i]["customer"], anchor="w",
        bg="white"
    ).place(x=20, y=(i + 1) * 30 + 120)
    tk.Label(
        self, width=20, text=results[i]["origin"], anchor="w", b
        g="white"
    ).place(x=180, y=(i + 1) * 30 + 120)
    tk.Label(
        self, width=20, text=results[i]["destination"], anchor="
        w", bg="white"
    ).place(x=320, y=(i + 1) * 30 + 120)
    tk.Label(
        self, width=20, text=results[i]["weight"], anchor="w", b
        g="white"
    ).place(x=480, y=(i + 1) * 30 + 120)

```

Fitur lihat penyimpanan barang digunakan untuk pengirim melihat barang pelanggan yang belum dikirim atau masih berada pada gudang port. Pengirim terlebih dahulu memilih port yang ingin dilihat barangnya. Setelah itu, muncul detail barang dari pelanggan yang belum dikirim sesuai dengan port yang dipilih.

4.7.3. Fitur Lihat Profil

Kode Program go-cargo.py

```

class AdminProfile(Page):
    def __init__(self, parent):
        # init page/ delete old page
        Page.__init__(self, parent)

        response = requests.get(f"{url}/users/{userEmail}")
        results = response.json()

        tk.Label(
            self,
            text="Detail Profile",
            width=20,
            bg="white",
            anchor="w",
            font=("bold", 20),
        ).place(x=20, y=60)

        tk.Label(self, text="Nama", width=20,
            bg="white", anchor="w").place(x=20, y=120)
        tk.Label(

```

```

        self, text=results["name"], width=20, bg="white", anchor="w").place(x=100, y=120)

        tk.Label(self, text="Email",
                  width=20, bg="white", anchor="w").place(x=20, y=150)
        tk.Label(
            self, text=results["email"], width=20, bg="white", anchor="w").place(x=100, y=150)

        tk.Label(
            self, text="Jenis Kelamin", width=20, bg="white", anchor="w"
        ).place(x=20, y=180)
        tk.Label(
            self, text=results["gender"], width=20, bg="white", anchor="w"
        ).place(x=100, y=180)

        tk.Label(self, text="Umur", width=20,
                  bg="white", anchor="w").place(x=20, y=210)

        tk.Label(
            self, text=results["age"], width=20, bg="white", anchor="w").place(x=100, y=210)

        tk.Label(
            self, text="Tambah Pengirim Baru", width=20, bg="white", font=("bold", 20)
        ).place(x=400, y=60)

        tk.Label(self, text="Nama Lengkap", width=20).place(x=400, y=120)
        self.entry_name = tk.Entry(self)
        self.entry_name.place(x=550, y=120)

        tk.Label(self, text="Email", width=20).place(x=400, y=150)
        self.entry_email = tk.Entry(self)
        self.entry_email.place(x=550, y=150)

        tk.Label(self, text="Password", width=20).place(x=400, y=180)
        self.entry_password = tk.Entry(self, show="*")
        self.entry_password.place(x=550, y=180)

        tk.Label(self, text="Jenis Kelamin", width=20).place(x=400, y=210)

        self.var_gender = StringVar()

        self.select_gender = tk.Radiobutton(
            self, text="Pria", padx=5, value="pria", variable=self.var_gender
        ).place(x=550, y=210)

```

```

self.select_gender = tk.Radiobutton(
    self, text="Wanita", padx=20, value="wanita", variable=self.var_
gender
).place(x=600, y=210)

tk.Label(self, text="Umur:", width=20).place(x=400, y=240)

self.entry_age = tk.Entry(self)
self.entry_age.place(x=550, y=240)

tk.Button(
    self,
    text="Tambah pengirim",
    width=20,
    bg="dodger blue",
    fg="white",
    command=lambda: Auth.handleRegister(self, "shipper"),
).place(x=400, y=280)

```

Pada fitur lihat profile, pengirim dapat melihat detail profilnya antara lain nama, email, jenis kelamin dan umur. Disini pengirim hanya bisa melihat profilnya saja dan tidak bisa merubah profilnya. Selain itu, pengirim dapat menambah pengirim baru dengan memasukkan nama, email, password, jenis kelamin dan umur.

4.8. Looping

Kode Program go-cargo.py

```

if __name__ == "__main__":
    app = Auth()
    app.mainloop()

```

Dalam pengoperasiannya program ini menggunakan sistem looping. Program akan terus berjalan selama user tidak menutupnya.

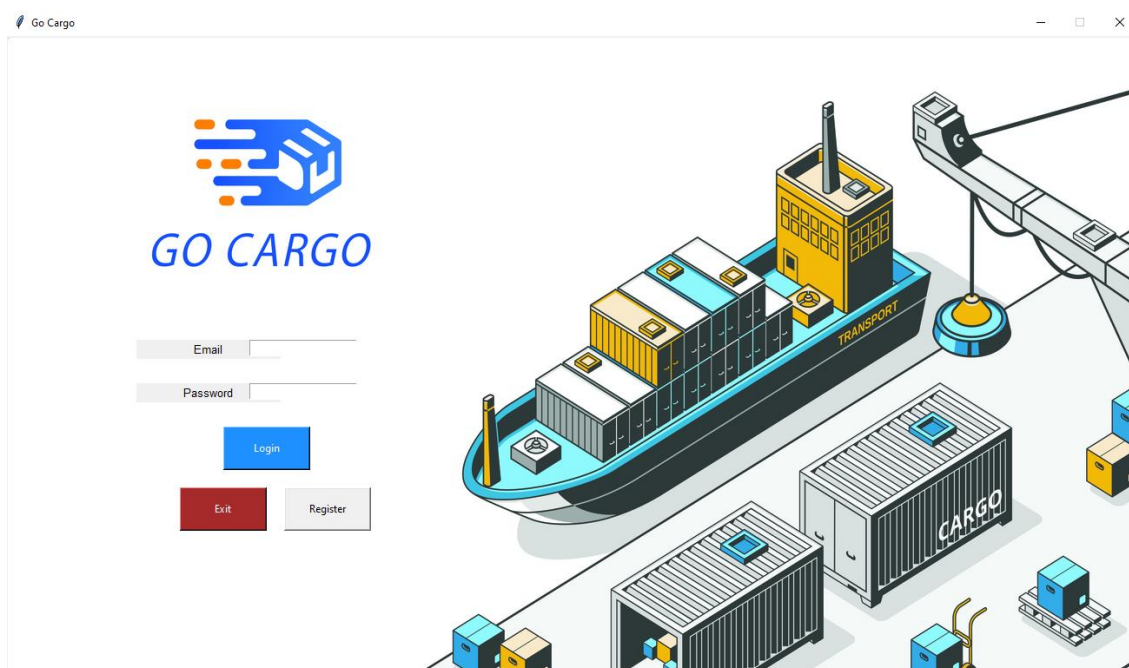
BAB V

HASIL DAN PEMBAHASAN

Pada bab ini kami akan mendeskripsikan hasil dari aplikasi yang kami buat berdasarkan pengoperasian pada setiap fiturnya yang akan dijelaskan pada sub bab berikut ini.

5.1. Untuk Users sebagai Customers

Pada tampilan awal akan menampilkan halaman login. Halaman ini akan menampilkan tiga pilihan menu yang dapat diakses oleh users yaitu login, register dan exit.



Gambar 4.1-1 Tampilan halaman login Go Cargo

5.1.1. Menu Login

Menu ini merupakan fitur yang pertama kali harus dilakukan pengguna agar dapat mengoperasikan aplikasi yang kami buat (dapat dilihat pada gambar 4.1-1).

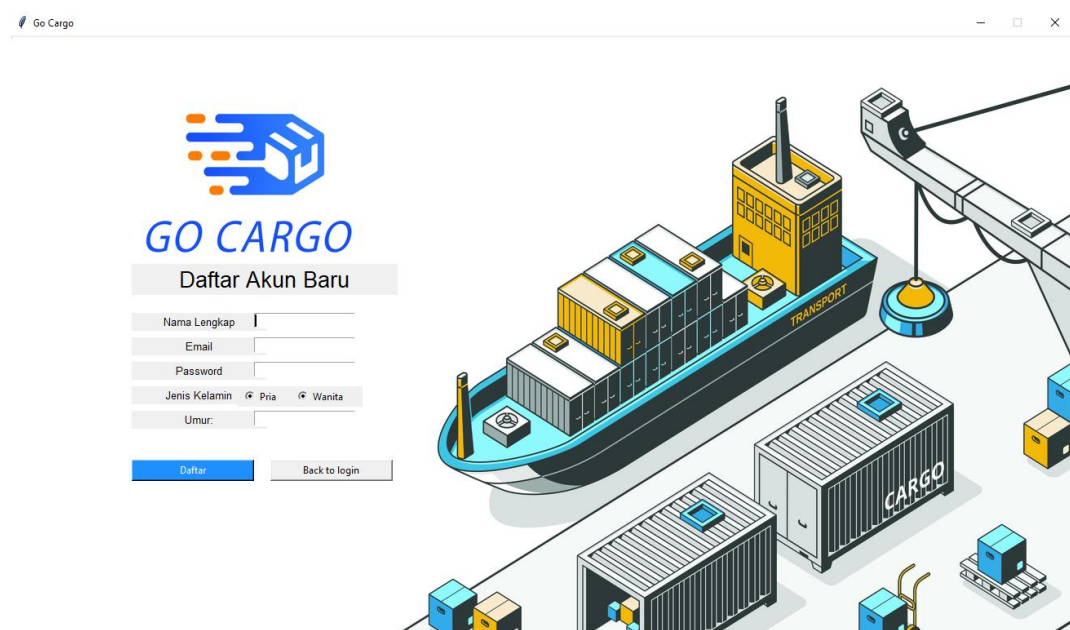
Menu login berfungsi untuk melakukan proses identifikasi user. Untuk cara pengoperasiannya cukup mudah yaitu dengan menginputkan email dan password pada akun yang sudah terdaftar pada system lalu klik 'login'. Apabila data login yang diinputkan itu benar. Maka, system akan mengirimkan notifikasi atau keterangan sukses dan begitu juga untuk sebaliknya. Kemudian, Apabila pengguna sudah pernah mengakses dan mengoperasikan aplikasi ini sebelumnya, pengguna bisa login menggunakan akun yang telah terdaftar sebelumnya.



Gambar 4.1.1-1 Contoh tampilan notifikasi sukses melakukan login

5.1.2. Menu Registrasi Akun

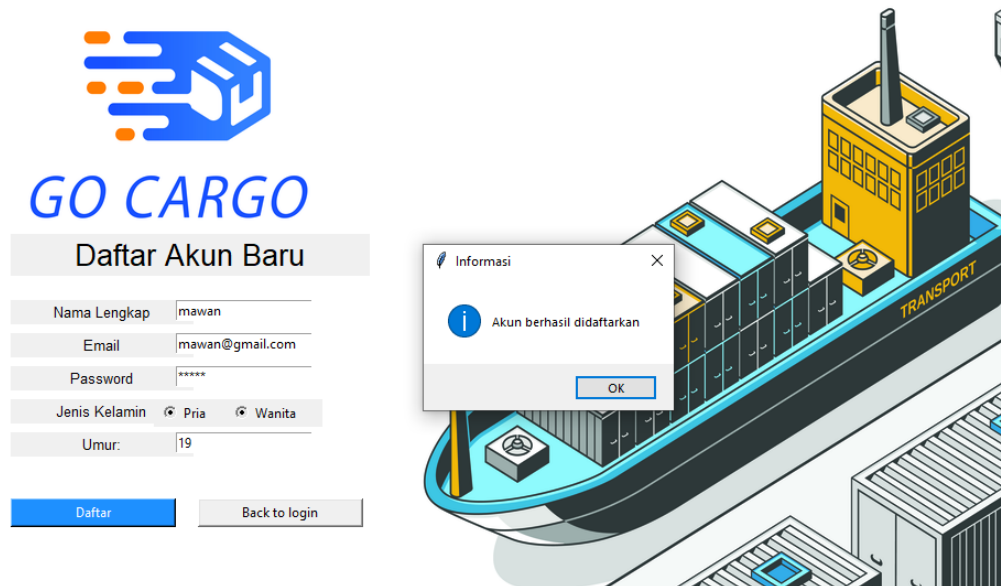
Menu registrasi ini bersifat opsional. User bisa mengabaikan fitur ini apabila sudah memiliki akun untuk login ke aplikasi. Apabila belum memiliki akun, maka user wajib menggunakan fitur ini untuk membuat akun baru.



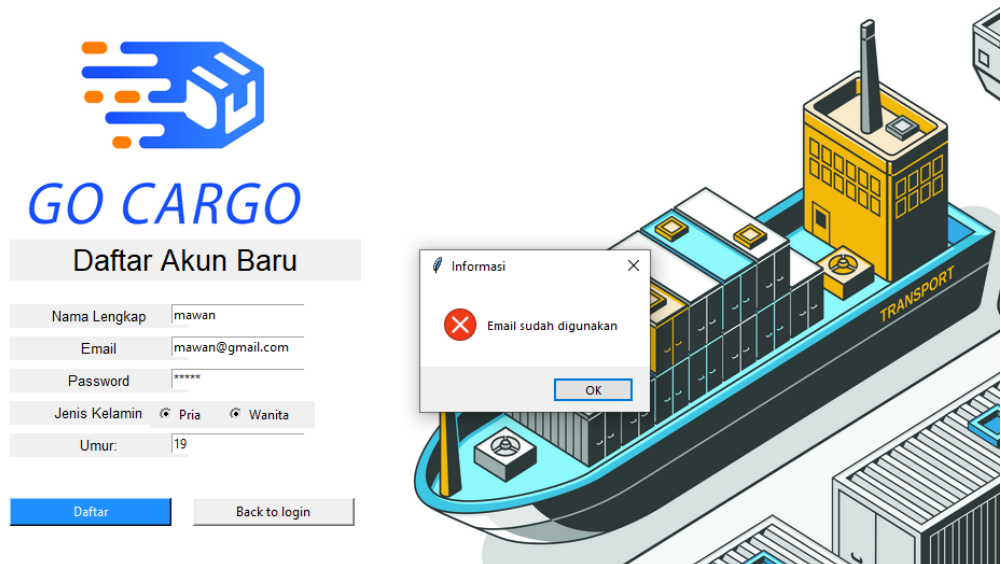
Gambar 4.1.2-1 Contoh tampilan halaman Register – Daftar Akun Baru

Untuk cara pengoperasiannya cukup mudah yaitu dengan mengisi form yang sudah disediakan yaitu nama lengkap, e-mail, password, jenis kelamin, dan umur lalu klik 'Daftar'. Maka akan muncul notifikasi atau keterangan bahwa akun telah berhasil didaftarkan. Satu e-mail hanya dapat digunakan untuk melakukan satu kali

pendaftaran. Apabila ada e-mail yang sama digunakan untuk mendaftar. Maka system akan menolak perintah tersebut dengan mengirimkan notifikasi bahwa data tersebut sudah terpakai. Apabila pengguna ingin membatalkan operasi ini, maka cukup klik 'Back to login' untuk kembali ke halaman login.



Gambar 4.1.2-2 Contoh tampilan notifikasi sukses melakukan pendaftaran akun



Gambar 4.1.2-3 Contoh tampilan notifikasi gagal melakukan pendaftaran akun

5.1.3. Menu Home atau Beranda

Fitur ini akan muncul untuk pertama kalinya setelah user berhasil melakukan login ke dalam aplikasi Go Cargo. Fitur ini berfungsi untuk melakukan permintaan pengiriman barang dari user sebagai customers. Untuk mengoperasikannya,

customers dapat menginputkan data-data pengiriman barang yang dibutuhkan. Customer dapat memilih kota asal, kota tujuan dan berat barang yang ingin dikirimkan lalu klik 'submit'. Maka, sistem secara otomatis akan menampilkan detail pengiriman dan detail tersebut akan terecord oleh system lalu diteruskan ke pihak admin Go Cargo untuk diproses lebih lanjut.

The screenshot shows a web interface for a customer menu. On the left is a dark sidebar with a 'Customer Menu' header and links for 'Home', 'Cek Pengiriman', 'Profile', and 'Exit'. The main content area has a form with three input fields: 'Pilih kota asal' (set to 'Jakarta Utara'), 'Pilih kota tujuan' (set to 'Banyuwangi'), and 'Berat (kg)' (set to '1'). A blue 'submit' button is to the right. Below the form, a 'Detail Pengiriman :' section displays the following information:

Kota Asal	Jakarta Utara
Kota Tujuan	Banyuwangi
Berat Barang	150
Biaya Pengiriman	23000
Estimasi Pengiriman	3 hari

Gambar 4.1.3-1 Contoh tampilan pengiriman barang pada halaman Beranda

5.1.4. Menu Cek Pengiriman

Fitur ini digunakan untuk melakukan pengecekan status pengiriman barang. Semua riwayat pengiriman barang customer akan ditampilkan pada fitur ini. Customer dapat memantau secara langsung perkembangan status pengirimannya, apakah barang yang telah diserahkan ke pihak Go Cargo sudah terkirim atau masih dalam proses pengiriman.

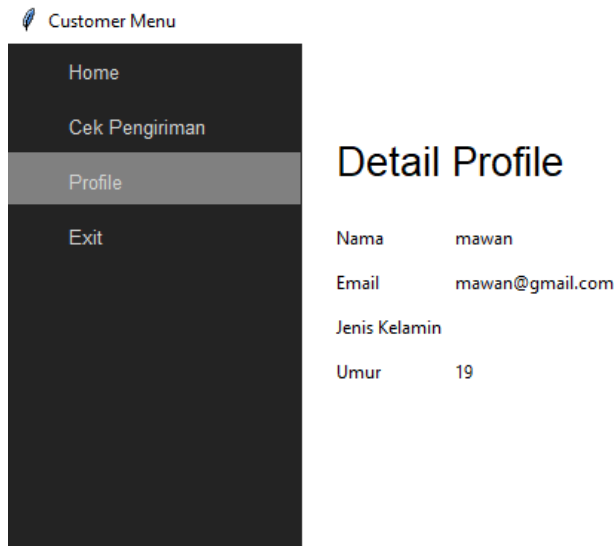
The screenshot shows the 'Cek Pengiriman' section of the Customer Menu. It displays a table with the following data:

pengirim	kota asal	kota tujuan	berat (kg)	status
varrel tantio	Jakarta Utara	Jakarta Utara	120	Sudah terkirim
varrel tantio	Jakarta Utara	Banyuwangi	150	Sudah terkirim
varrel tantio	Jakarta Utara	Semarang	750	Sudah terkirim
varrel tantio	Jakarta Utara	Makassar	450	Sudah terkirim
varrel tantio	Jakarta Utara	Pontianak	500	Sudah terkirim
varrel tantio	Jakarta Utara	Surabaya	300	Sudah terkirim
-	Jakarta Utara	Medan	1500	Proses pengiriman
-	Jakarta Utara	Banyuwangi	900	Proses pengiriman
-	Jakarta Utara	Jayapura	1800	Proses pengiriman

Gambar 4.1.4-1 Contoh tampilan dari menu cek pengiriman barang

5.1.5. Menu Profile

Fitur ini sangat simple yakni untuk menampilkan detail profile dari customer seperti nama, e-mail, umur dan jenis kelamin. Untuk saat ini, hanya itu yang dapat ditampilkan. Kedepannya akan diupdate untuk pengembangan lebih lanjut.



Gambar 4.1.5-1 Contoh tampilan dari menu profile

5.1.6. Exit

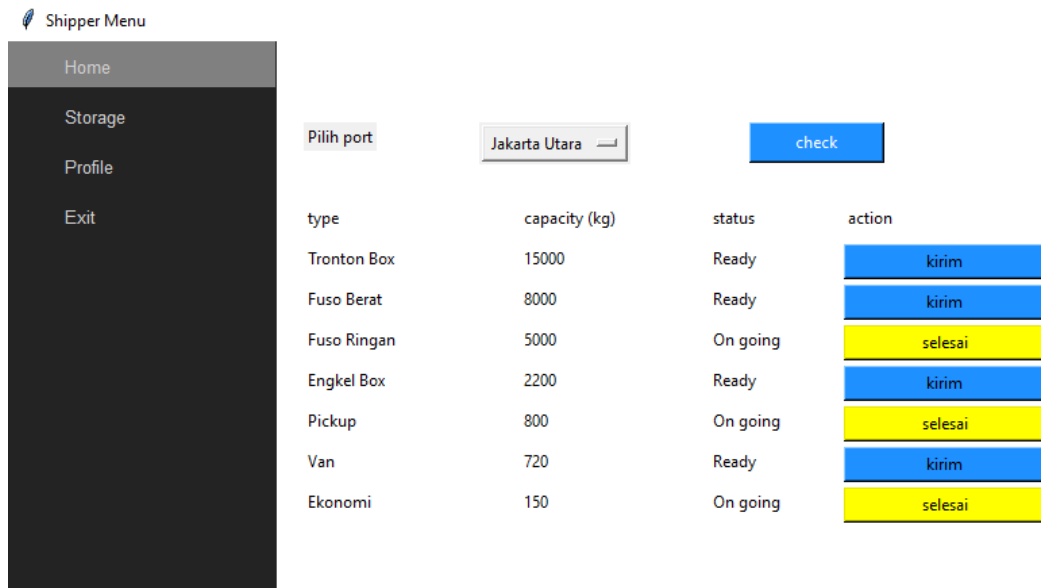
User keluar dari menu utama aplikasi. Secara otomatis sistem akan mengembalikan user ke halaman login aplikasi Go Cargo (dapat dilihat pada gambar 4.1-1).

5.2. Untuk Users sebagai Admin Pengirim

Fitur ini merupakan fitur lanjutan setelah pengguna mendapatkan hak akses untuk masuk ke dalam aplikasi. Fitur ini menampilkan menu utama yang terdapat di dalam aplikasi ini. Pada tampilan awal akan menampilkan halaman login yang sama dengan halaman login milik customers (dapat dilihat pada gambar 4.1-1). Akan tetapi, akses sebagai admin hanya dapat digunakan apabila memiliki e-mail khusus untuk admin sebagai pengirim

5.2.1. Menu Home atau Beranda

Fitur ini akan muncul untuk pertama kalinya setelah user atau admin pengirim berhasil melakukan login ke dalam aplikasi Go Cargo.



Gambar 4.2.1-1 Contoh tampilan dari menu home atau pengiriman barang

Fitur pada menu home memiliki fungsi utama untuk mengirimkan barang sesuai detail pengiriman dari customers. Go Cargo memiliki beberapa armada dengan kapasitas pengiriman yang berbeda. Cara pengoperasian pengiriman barang pada aplikasi ini melibatkan menu storage (yang akan dijelaskan di sub bab 4.2.2) dan fitur pada menu home. Sebagai contoh pengiriman barang dari port atau kota asal Jakarta Utara, cara melakukannya sebagai berikut.

1. Sebelum melakukan pengiriman barang, admin terlebih dahulu melakukan pengecekan pada menu storage. Ada 3 pengiriman ke tiga kota tujuan yang berbeda dari customer yang sama, yang perlu diselesaikan (dapat dilihat pada gambar 4.2.2-1).
2. Setelah selesai melakukan pengecekan di menu storage, Admin pengirim akan mengerti barang apa saja yang perlu dikirimkan. Selanjutnya, admin dapat memproses pengiriman dengan menggunakan fitur yang ada di menu home. (Lihat pada gambar 4.2.2-1). Karena ada pengiriman dengan kota tujuan yang searah. Maka, beberapa barang dapat dikirimkan secara bersamaan agar lebih efektif atau menghemat kapasitas dari armada. Ekspedisi ke timur, Banyuwangi dan Jayapura memiliki total berat barang sebesar 2700 kg. Maka, pengiriman dapat diselesaikan menggunakan armada Fuso Ringan yang memiliki kapasitas 5000 kg. Kemudian, Ekspedisi ke barat, Medan memiliki total berat barang

sebesar 1500 kg. Maka, pengiriman dapat diselesaikan menggunakan armada Engkel Box yang memiliki kapasitas sebesar 2200 kg.

- Setelah mengerti barang dan armada yang akan digunakan. Admin dapat memproses pengiriman dengan klik ‘kirim’ pada armada yang akan digunakan. Maka, secara otomatis system akan menampilkan notifikasi bahwa barang telah dikirim dan status barang menjadi ‘on going’ serta status action akan berubah menjadi warna kuning. Apabila, admin sudah mendapatkan laporan dari kurir bahwa barang sudah diterima di kota tujuan. Admin dapat mengklik ‘selesai’. Maka, secara otomatis system akan menampilkan notifikasi bahwa barang selesai dikirim dan status action akan berubah warna menjadi biru yang menandakan bahwa armada akan siap untuk digunakan kembali.

5.2.2. Menu Storage

Secara umum, menu ini menampilkan data-data terkait detail pengiriman dan barang yang perlu diproses atau dikirimkan. Data pengiriman dari setiap kota berbeda-beda. Admin dapat mengakses data-data pengiriman sesuai kebutuhan dengan cara menentukan port atau kota asal lalu klik ‘check’. Maka, secara otomatis system akan menampilkan seluruh data sesuai perintah yang telah diberikan.

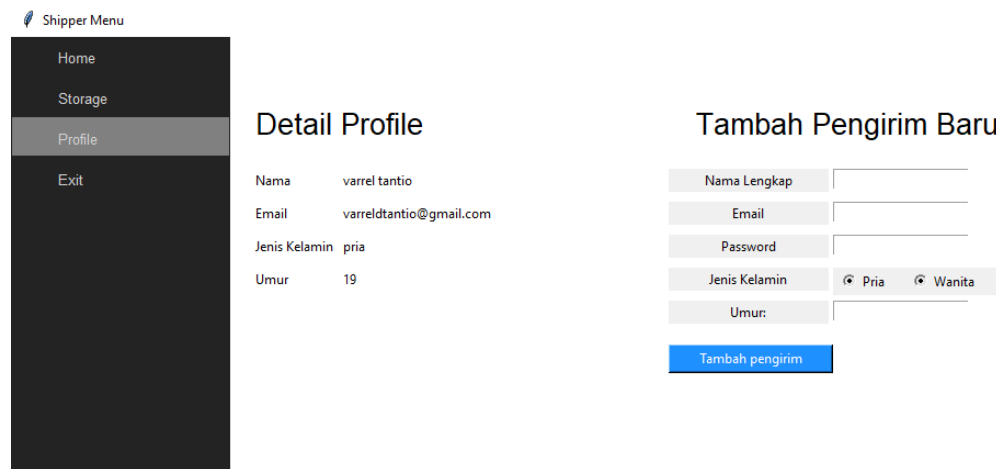
customer	kota asal	kota tujuan	berat (kg)
mawan	Jakarta Utara	Medan	1500
mawan	Jakarta Utara	Banyuwangi	900
mawan	Jakarta Utara	Jayapura	1800

Gambar 4.2.2-1 Contoh tampilan dari menu storage

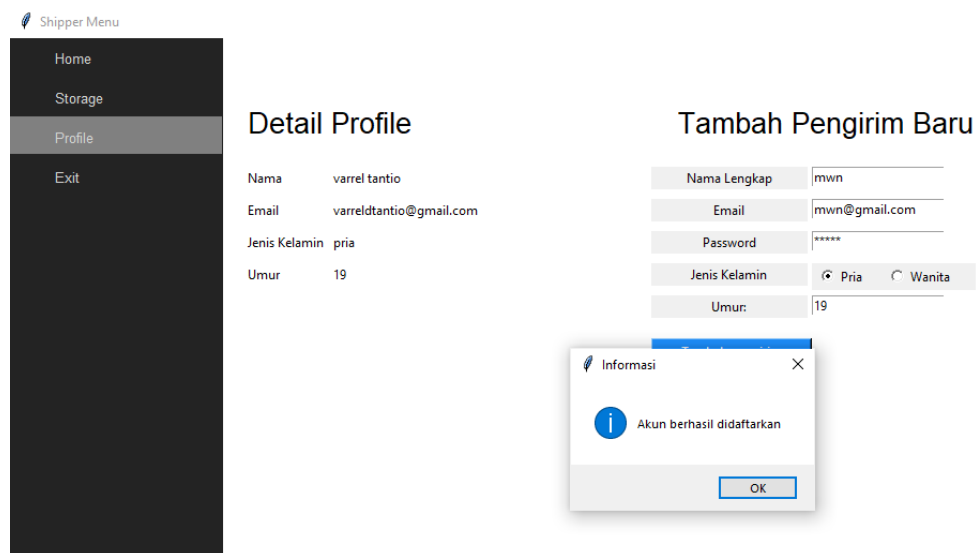
5.2.3. Menu Profile

Menu profil ini digunakan untuk melihat detail profile dari pengirim seperti nama, e-mail, umur dan jenis kelamin. Selain itu, menu ini dapat digunakan untuk menambahkan akun pengirim baru dengan cara mengisi data-data yang dibutuhkan pada form yang sudah disediakan di sebelah kanan dari detail profile.

Setelah selesai mengisi data-data yang diperlukan, klik ‘tambah pengirim’. Maka, secara otomatis sistem akan menampilkan notifikasi bahwa akun telah berhasil didaftarkan.



Gambar 4.2.3-1 Contoh tampilan dari menu profile



Gambar 4.2.3-2 Contoh tampilan notifikasi sukses menambahkan pengirim baru

5.2.4. Exit

User sebagai admin pengirim akan keluar dari menu utama aplikasi. Secara otomatis sistem akan mengembalikan user ke halaman login aplikasi Go Cargo (dapat dilihat pada gambar 4.1-1).

DAFTAR PUSTAKA

- [1] <https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/> [diakses pada tanggal 03 April 2021]
- [2] <https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/> [diakses pada tanggal 03 April 2021]
- [3] <https://www.geeksforgeeks.org/rat-in-a-maze-backtracking-2/> [diakses pada tanggal 03 April 2021]
- [4] <https://mti.binus.ac.id/2017/11/28/algoritma-dijkstra/> [diakses pada tanggal 04 April 2021]
- [5] Wiyanti, Dian Tri. 2013. Algoritma Optimasi Untuk Penyelesaian Travelling Salesman Problem. Jurnal Tranformatika, 11(1), 1-4. [diakses pada tanggal 04 April 2021]

LAMPIRAN

Berikut link presentasi dari demo aplikasi kami.

Link Presentasi Youtube : <https://youtu.be/sD-sgeG6jTg>