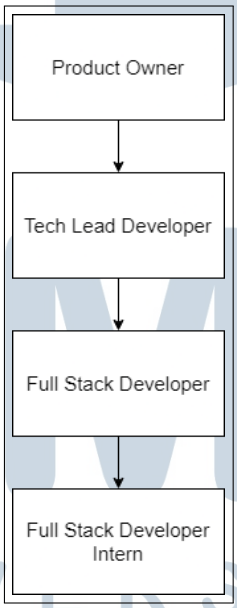
BAB 3
PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Organisasi

Pada proses kerja magang di PT. Siloam International Hospitals, penulis diberi tanggung jawab sebagai *Fullstack Developer*. Selama proses kerja magang, Kak Stevani Lilyani Saputri selaku Tech Lead Developer dari perusahaan memberikan bimbingan dan pengawasan secara langsung dibantu dengan arahan oleh developer lainnya yaitu Kak Vincensa Regina.

Untuk sarana komunikasi, para developer, serta pengguna internal menggunakan Jira dan Whatsapp. Awal pengerjaan task, developer diberikan tiket Jira dimana terdapat *scope task* beserta penjelasannya. Jika proses pengerjaan sudah selesai, developer akan melakukan *request merge* dimana akan dilakukan pengecekan oleh tim *product* apakah sesuai ekspektasi.



Gambar 3.1. Kedudukan dan Organisasi Pelaksanaan Magang

3.2 Tugas yang Dilakukan

Tugas yang dilakukan sebagai *Fullstack Developer* pada proses magang di PT. Siloam International Hospitals adalah sebagai berikut:

1. Meningkatkan *data engineering* pada *website* perusahaan menggunakan *Elastic Stack*
2. Melaksanakan tugas *IT* Operasional apabila terdapat kendala yang membutuhkan penanganan segera
3. Melakukan *fixing bug* dan *error* pada *website*

3.3 Uraian Pelaksanaan Magang

Pelaksanaan kerja magang diuraikan seperti pada Tabel 3.1 dan 3.2.

Tabel 3.1. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	**Pekerjaan yang dilakukan**
1	Melakukan instalasi tools yang dibutuhkan dan memahami API - API yang ada pada *website*
2	Mempelajari *framework* Vuejs serta mengerjakan *component* cari dokter pada halaman detail rumah sakit
3	Membuat *component* list dokter pada halaman cari dokter dan mengimplementasikan *Article Structured Data* pada halaman artikel di *website*
4	Mempelajari *Elastic* Stack terutama fitur *Elastic Search* dan *Kibana* dalam melakukan pengumpulan data dan mengolah data menjadi visualisasi pada *Kibana*
5	Mengerjakan *backlog* dalam mengolah data dengan memasang *tracker* dan membuat visualisai data
6	Memasang *tracker* untuk funneling health service dan funneling doctor
7	Memasang *tracker* untuk funneling artikel serta menambahkan variabel sesuai permintaan dari tim product
8	Melakukan *fix bug* pada *tracker* yang dipasang serta melakukan update *sheet excel* tentang list *tracker* yang tersedia
9	Mengerjakan *issue* dan melakukan *fix tracker* serta melakukan *refactor code* atau optimisasi kode yang sudah dibuat

Lanjut pada halaman berikutnya

Tabel 3.2. Pekerjaan yang dilakukan tiap minggu selama pelaksanaan kerja magang

Minggu Ke -	Pekerjaan yang dilakukan
10	Mengerjakan *side bar menu* pada proyek *Corporate Governance* dan melakukan cek data pada visualisasi di *Kibana*
11	Mengerjakan *layout table* pada proyek *Corporate Governance* dan melakukan cek data pada visualisasi di *Kibana*
12	Melakukan integrasi API pada halaman *Corporate Governance*
13	Melakukan penyesuaian API pada halaman *Corporate Governance* dan *check* ulang visualisasi di *Kibana*
14	Melakukan *fix* tampilan halaman *Corporate Governance*
15	Melakukan *layouting* halaman pada proyek Healthpedia
16	Melakukan integrasi API pada proyek Healthpedia dan mengerjakan task SEO *(Search Engine Optimizer*
17	Mengerjakan *Mega Menu Navbar* dan *fix* halaman Healthpedia
18	Melakukan *fix bug* dan API pada proyek Healthpedia

3.4 Perancangan

Pada saat melakukan diskusi dengan tim *product* dan *developer*, tim *developer* diminta untuk membuat visualisasi *funneling* pada Kibana. Tim *product* memberikan daftar *funneling* yang perlu dibuat beserta penjelasan detailnya melalui *Google Sheets*. Setiap minggu pada hari rabu, tim *product* dan tim *developer* akan melakukan *daily run-through* untuk melihat *progress*.

Funneling adalah gambaran alur perjalanan *user* mulai awal hingga akhir atau tercapainya tahap terakhir pada suatu proses. *Funneling* biasa disebut *user journey* yang dibuat untuk memudahkan bisnis dalam menjangkau pelanggannya. Kegunaan *funneling* adalah untuk meningkatkan produktivitas bisnis terutama dalam proses penjualan.

Pembuatan visualisasi sebuah *funneling* memerlukan data yang dikumpulkan menggunakan *tracker*. *Tracker* dipasang pada beberapa bagian *website* yang diperlukan untuk membuat visualisasi. Input data berasal dari aktivitas *user* seperti memilih salah satu dokter. Data yang diterima dari aktivitas *user* dikirim menggunakan API Kibana. Data yang sudah dikumpulkan diolah menjadi sebuah *table* dengan *columns* yang sudah ditentukan oleh tim *product*

Daftar visualisasi yang harus dibuat ada sebanyak 30 visualisasi. Berikut

merupakan beberapa visualisasi berupa *funneling* dan *rank* :

1. *Funneling Health Service*

   Funneling Health Service adalah alur perjalanan user dalam membeli paket atau *Health Service* di *website*. *User* meminta visualisasi yang dibuat dibedakan dari nama paket yang dibeli oleh *user*. Jadi dari visualisasi tersebut, tim bisa mengetahui paket mana yang paling sering dilihat atau dibeli oleh *user*.

2. *Funneling Health Service (User Age)*

   Funneling Health Service (User Age) adalah visualisasi yang menunjukkan jumlah *range age user* yang membeli paket pada *website*. Jadi dari visualisasi tersebut, tim bisa mengetahui *range user* yang sering membeli paket lewat *website* siloam.

3. *Funneling Doctor*

   Funneling Doctor adalah alur perjalanan user dalam membuat perjanjian bertemu dokter di *website*. *User* meminta visualisasi yang dibuat dibedakan dari nama dokter. Jadi dari visualisasi tersebut, tim bisa mengetahui dokter yang sering dilihat atau membuat janji pertemuan dengan *user*.

4. *Rank Search Filter*

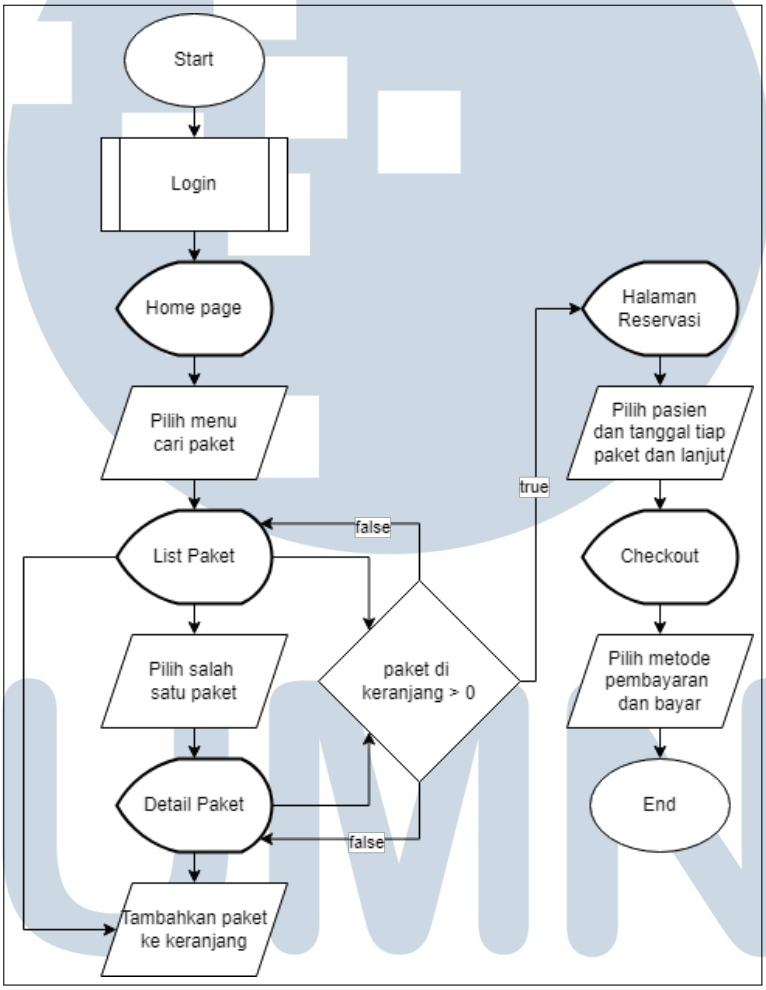
   Rank Search Filter adalah salah satu fitur yang ada pada halaman cari dokter. *Filter* yang disediakan adalah spesialis dokter, hari tersedia, waktu yang tersedia, lokasi, dan tipe konsultasi. Visualisasi ini bertujuan untuk mengetahui *filter* mana yang paling membantu *user* dalam menemukan dokter.

5. *Rank Patient Profile*

   Rank Patient Profile adalah tipe pasien yang dipilih oleh *user* saat membuat janji pertemuan dengan dokter. Ketika *user* membuat janji pertemuan dengan dokter, *user* dapat memilih akun tersebut, memilih pasien yang sudah pernah dibuat, atau membuat pasien baru disaat membuat janji pertemuan tersebut. Jadi dari visualisasi tersebut, tim dapat mengetahui tipe pasien yang dipilih paling banyak saat membuat janji pertemuan dengan dokter.

Pada saat membuat visualisasi *funneling*, dibutuhkan flowchart untuk mengetahui alur dari *funneling* tersebut. Pada pembuatan visualisasi, dibuatlah *flowchart system* dan *mockup* visualisasi.

3.4.1 Flowchart *Funneling Health Service*



Gambar 3.2. Flowchart *Funneling Health Service*

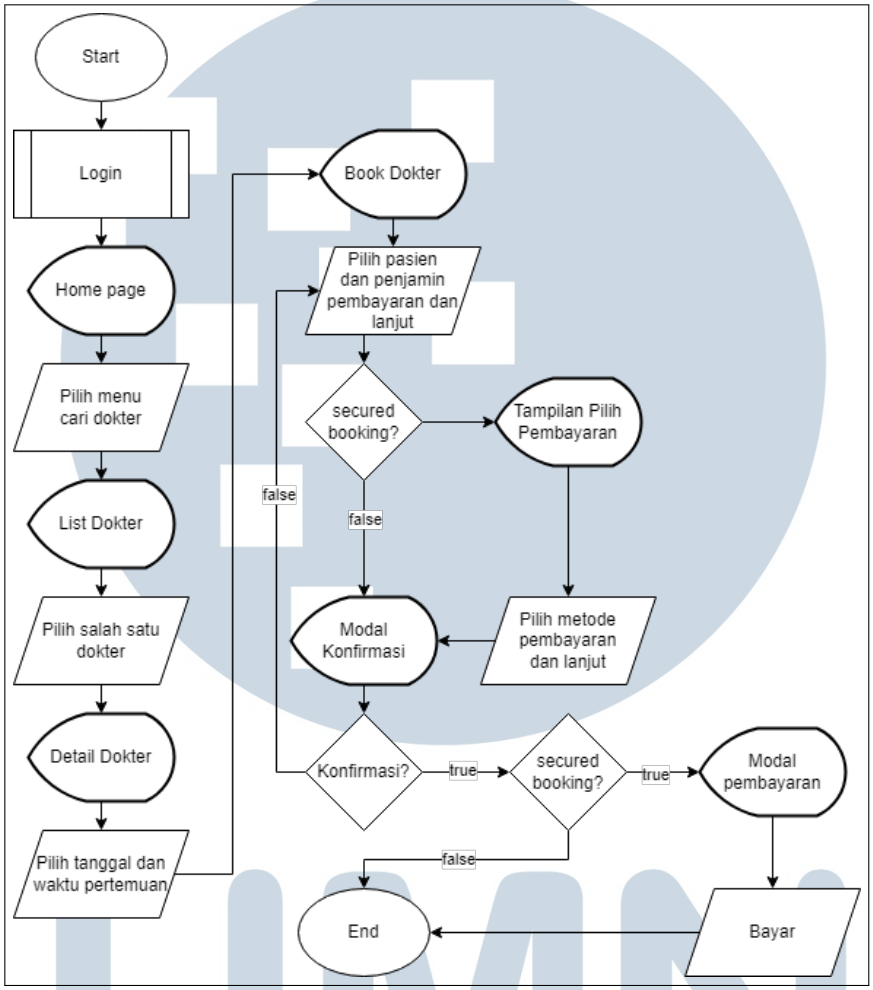
Flowchart berikut adalah alur *user* untuk memesan paket dimana paket adalah layanan dari rumah sakit sebagai contoh Paket Cek Fungsi Ginjal yaitu pasien melakukan cek fungsi ginjalnya. Pada flowchart ini, alur dari *funneling health service* pertama kali yaitu *user* harus melakukan login dahulu. Jika *user* tidak login, *user* hanya memiliki akses untuk melihat daftar paket dan buka paket detail. *User* dapat memasukkan paket ke keranjang pada saat melihat paket atau saat buka paket detail. Jika *user* telah memilih paket yang dia mau, *user* dapat

membuat reservasi terhadap paket yang dibeli dengan memilih data pasien, tanggal reservasi untuk setiap paket yang dibeli. Tahap selanjutnya adalah *user* memilih metode pembayaran dan melakukan pembayaran.

Tiap proses diatas sistem akan mengumpulkan data nama paket dalam *properties* dan dikirim sebagai *event*. Sebagai contoh dari melihat rincian paket, menambahkan paket ke keranjang, sampai proses selesai. Dengan melakukan pengumpulan data pada setiap proses, *developer* dapat membuat visualisasi *funneling* dari pembelian paket atau *Health Service*.

Pada perancangan, terdapat *Funneling Health Service (User Age)* dimana alur perjalanan yang dilakukan sama dengan yang telah dijelaskan diatas. Untuk membuat visualisasi *Funneling Health Service (User Age)*, sistem mengumpulkan dengan *event* yang sama, namun menambahkan *range user* dalam *properties*.

3.4.2 Flowchart *Funneling Doctor*

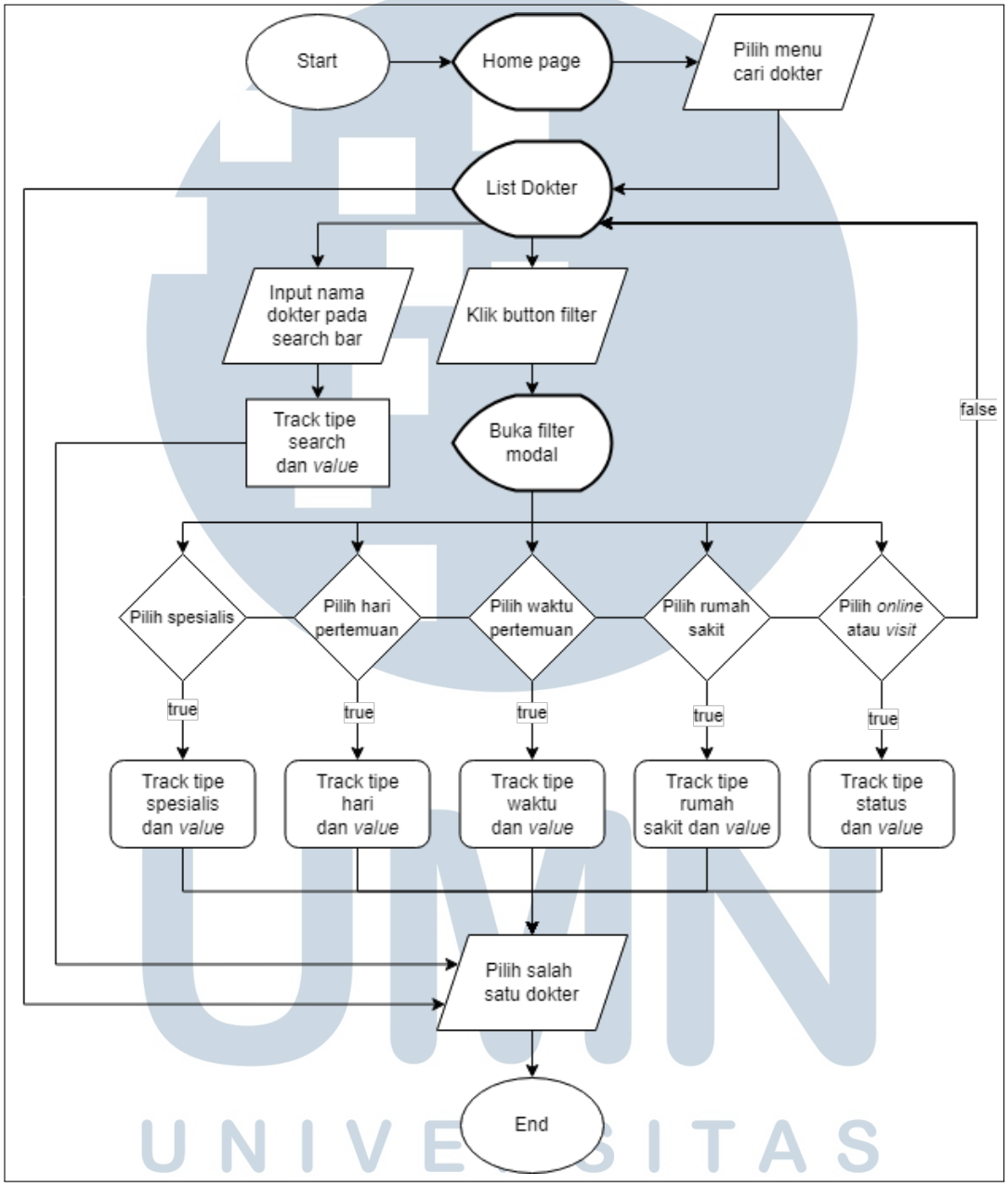


Gambar 3.3. Flowchart *Funneling Doctor*

Flowchart berikut adalah alur *user* untuk membuat janji pertemuan dengan dokter. Pada flowchart ini, alur dari *funneling doctor* adalah *user* melakukan login dahulu. Jika *user* tidak melakukan login, *user* hanya dapat melihat daftar dokter dan detail dokter tersebut. Jika *user* telah menemukan dokter yang ia mau buat janji pertemuan, *user* akan memilih rumah sakit atau klinik dan waktu untuk melakukan pertemuan dengan dokter. Tahap selanjutnya adalah *user* memilih data pasien serta penjamin pembayaran. Jika dokter yang dipilih *user* memiliki label *"Secured Booking"*, maka *user* harus melakukan pembayaran dahulu dengan keuntungan pertemuan tidak akan berganti jadwal atau ditunda. Jika dokter tidak bertipe *"Secured Booking"*, *user* cukup melakukan konfirmasi untuk melakukan pertemuan dengan dokter.

Tiap proses diatas sistem akan mengumpulkan data nama dokter dalam *properties* dan dikirim sebagai *event*. Namun terdapat pembeda yaitu jika tipe dokter *secured booking* atau tidak. Pembeda ini tidak akan membedakan *event* tersebut, tetapi yang membedakan adalah data yang dikumpulkan dimana yang membedakan adalah *properties* bool yang menyatakan apakah dokter yang sedang dibuat janji pertemuan bertipe *secured booking* atau tidak.

3.4.3 Flowchart *Rank Search Filter*

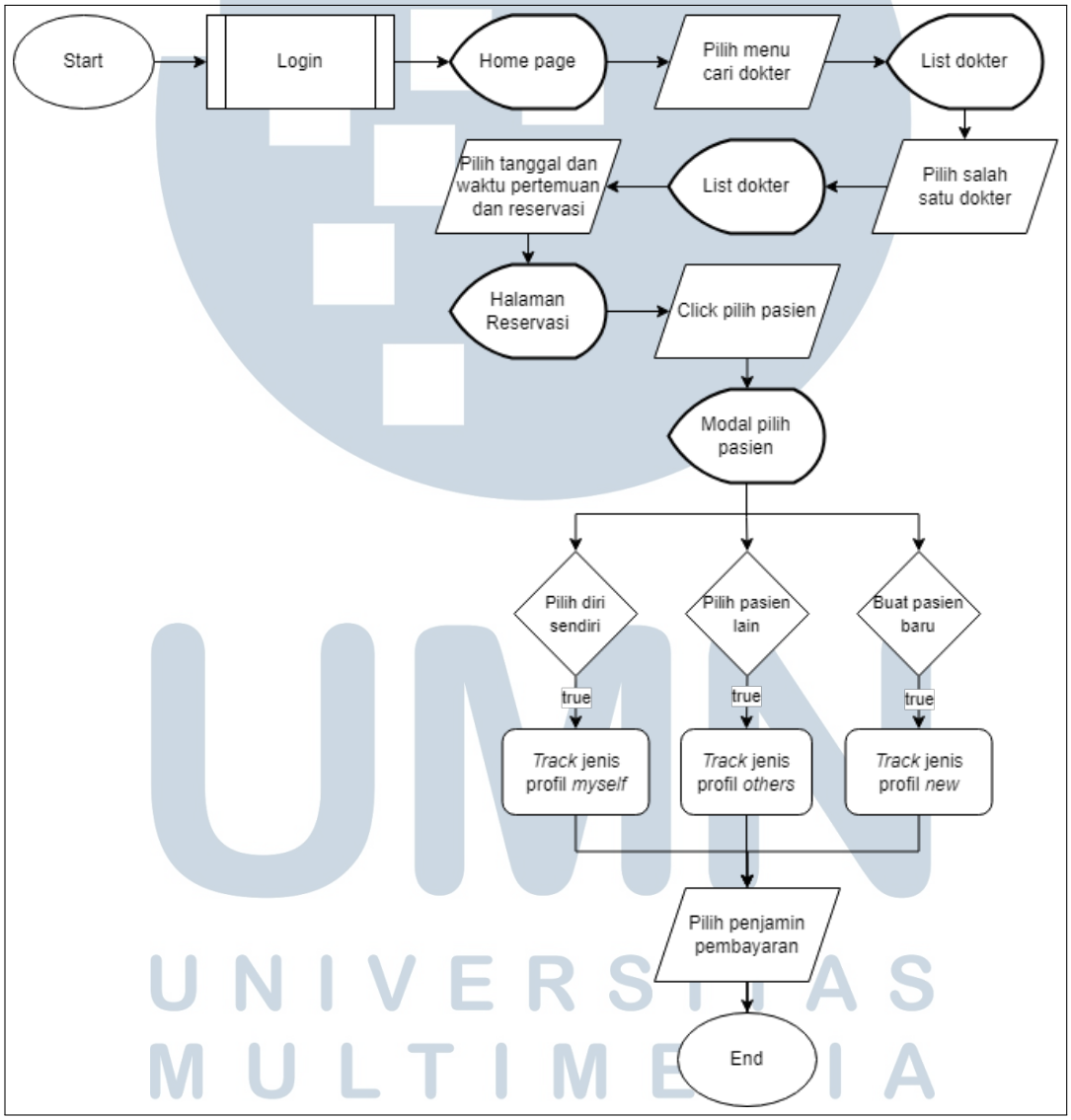


Gambar 3.4. Flowchart *Rank Search Filter*

Flowchart berikut adalah alur *user* untuk menggunakan fitur *search* dan *filter* pada mencari dokter. Untuk menggunakan fitur ini, *user* tidak diwajibkan untuk melakukan login dahulu sehingga dari *homepage* user bisa langsung ke cari dokter untuk melihat daftar dokter yang ada. Pada halaman cari dokter, terdapat *search bar* dan tombol *filter* untuk membuka modal *filter*. Dalam modal tersebut, *user* dapat

memanfaatkan *filter* secara spesialis, hari, waktu, rumah sakit, dan berdasarkan *online* atau *visit*. Tiap *user* menggunakan *search* atau salah satu *filter*, sistem akan mengumpulkan data tipe fitur yang dipakai dan *value* dalam sebuah *properties* dan dikirim sebagai *event*.

3.4.4 Flowchart *Rank Patient Profile*



Gambar 3.5. Flowchart *Rank Patient Profile*

Flowchart berikut adalah alur untuk menentukan tipe pasien yang dipilih saat membuat janji temu dengan dokter. Pada flowhcart ini, *user* harus melakukan login dahulu karena *user* harus membuat janji pertemuan dengan dokter. Tahap

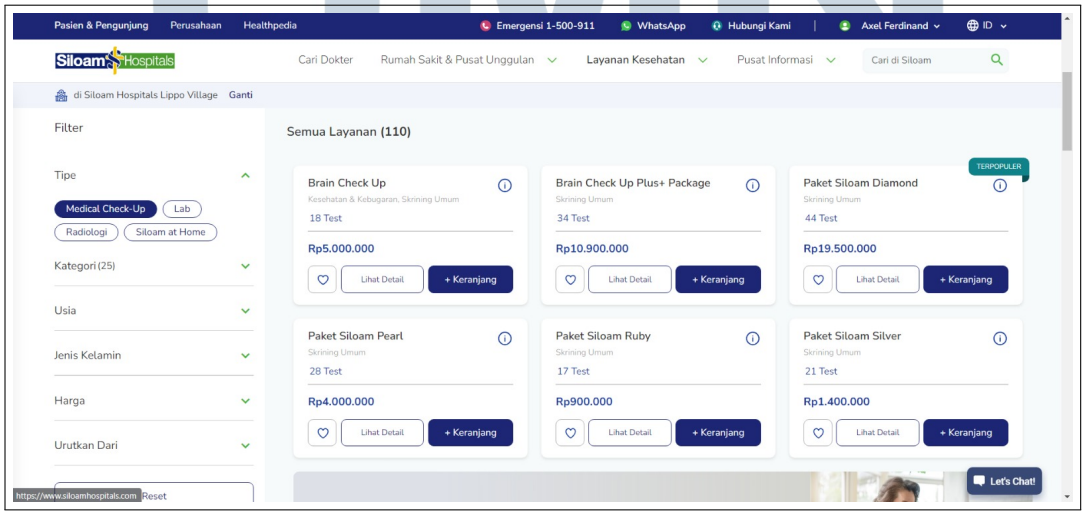
selanjutnya adalah ke halaman cari dokter dan pilih salah satu dokter dan buka halaman detail dokter. Pilih rumah sakit, tanggal, dan waktu untuk membuat janji pertemuan dan buat reservasi. Pada saat reservasi, *user* harus memilih pasien yang akan berkonsultasi dengan dokter. Terdapat 3 tipe pasien yaitu *myself* ketika *user* memilih diri sendiri, *others* ketika pasien memilih pasien yang sudah pernah dibuat, dan *new patient* ketika *user* membuat pasien baru disaat membuat janji pertemuan dengan dokter. Sistem akan mengumpulkan data tipe pasien pada *properties* dan mengirim dalam sebuah *event* setelah memilih pasien.

3.5 Implementasi

Pada implementasi *data engineering* menggunakan *Elastic Stack*, data yang dikumpulkan merupakan data yang berasal dari *front-end* atau langsung dari *user*. Implementasi dilakukan dengan mengirim data-data yang mau dikumpulkan dan dikirim ke *Elastic* sebagai sebuah *event*. Sebagai contoh, tim *product* ingin melihat paket apa yang paling banyak dilihat oleh *user*. Kami memasang *tracker* tiap sebuah paket pada *website* dibuka dengan nama *event* ”VIEW_PRODUCT” serta mengirimkan data berupa nama paket tersebut.

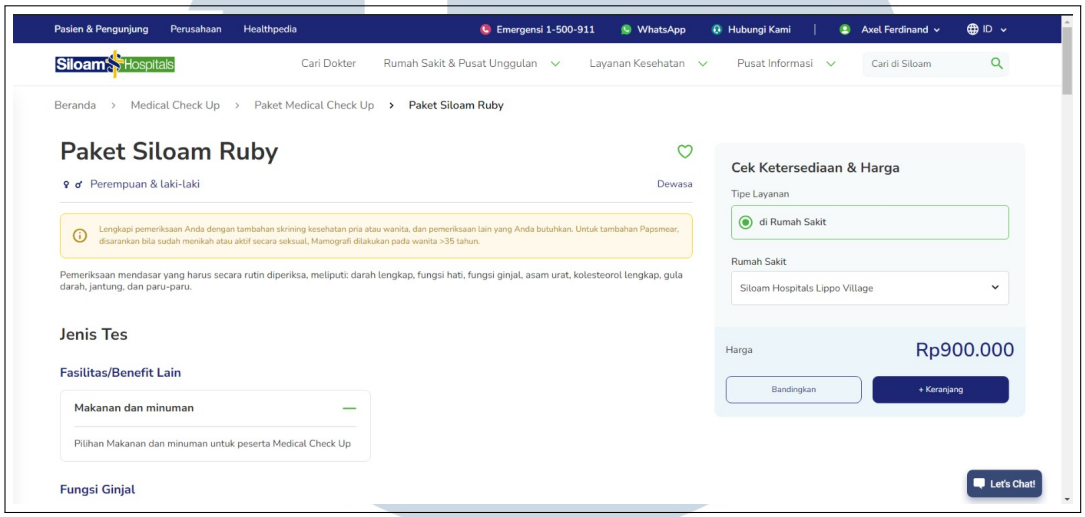
Berikut adalah implementasi dan hasil visualisasi serta penjelasan sesuai dengan perancangan yang telah dilakukan. Hasil visualisasi ditampilkan pada *range* 1 minggu yaitu dari 1 May sampai 7 Mai.

3.5.1 *Funneling Health Service*



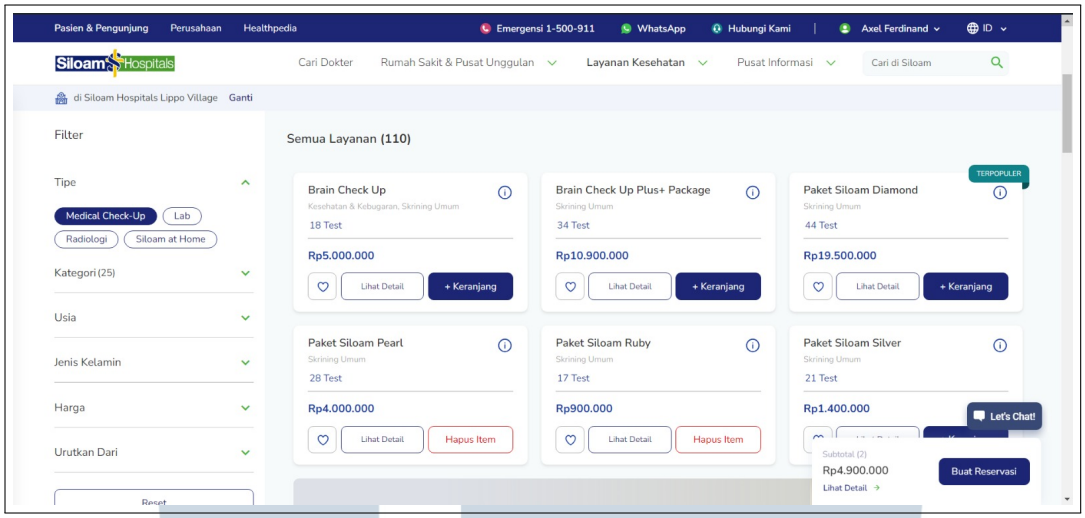
Gambar 3.6. Tampilan melihat semua paket

Pada visualisasi *funneling health service*, perjalanan *user* diawali dengan melakukan login dahulu dan membuka halaman lihat semua paket seperti pada Gambar 3.6. Pada tampilan ini, sistem melakukan pengumpulan data nama paket sebagai *properties* dan dikirim pada sebuah *event* yang dikirim ke *Elastic*. Jadi pada *event* ini, berisi paket-paket yang ditampilkan pada halaman ini. Tahap selanjutnya adalah *user* memilih salah satu paket dan lihat detail paket tersebut.

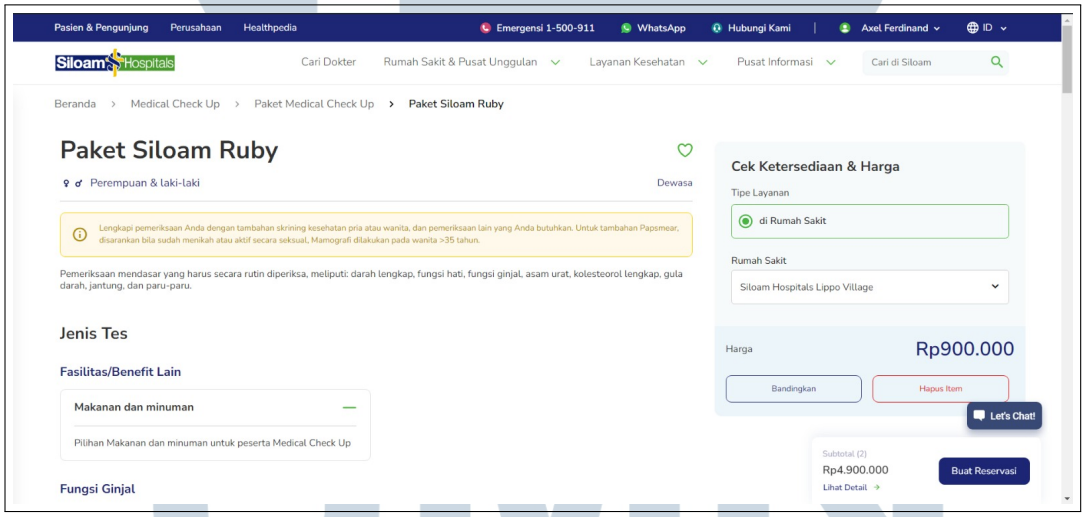


Gambar 3.7. Tampilan *View Package*

Pada saat *user* membuka halaman detail paket seperti di Gambar 3.7, sistem akan mengirimkan data nama paket yaitu Paket Siloam Ruby dalam *properties* item_name sebagai *event* ke *Elastic*. Pada halaman lihat daftar paket dan detail paket, *user* dapat menambahkan paket ke keranjang untuk reservasi paket tersebut. Pada saat menambahkan paket ke keranjang, sistem akan mengirimkan sebuah *event* dengan nama paket sebagai *properties*.

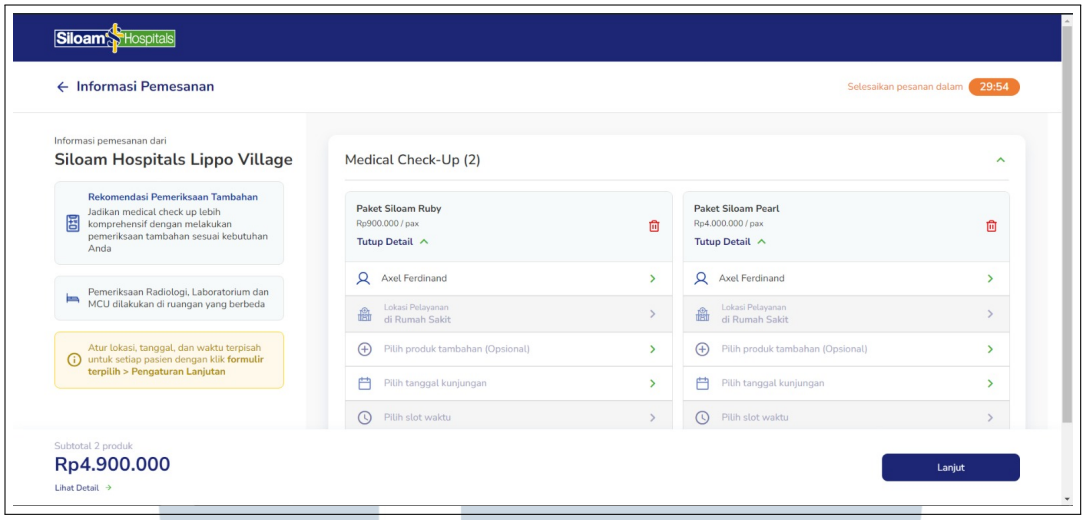


Gambar 3.8. Tampilan menambahkan paket pada halaman daftar paket



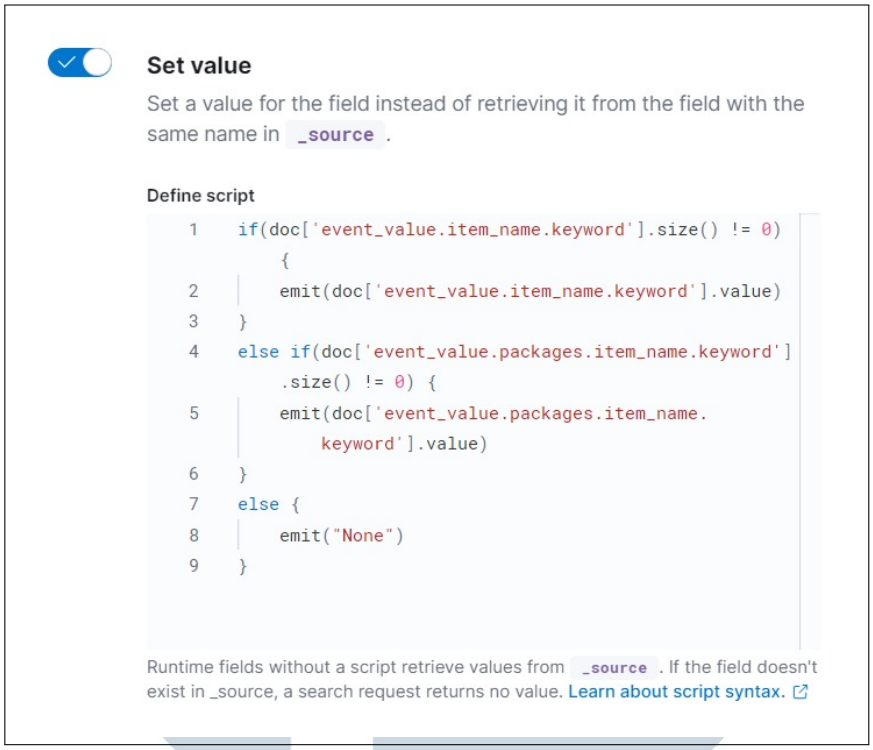
Gambar 3.9. Tampilan menambahkan paket pada halaman detail paket

Pada Gambar 3.8 adalah tampilan jika *user* telah menambahkan salah satu paket ke keranjang di halaman daftar paket. Untuk Gambar 3.9 adalah *user* yang menambahkan paket pada halaman detail paket. Langkah selanjutnya adalah membuat reservasi paket dengan menekan *button* buat reservasi pada bawah kanan.



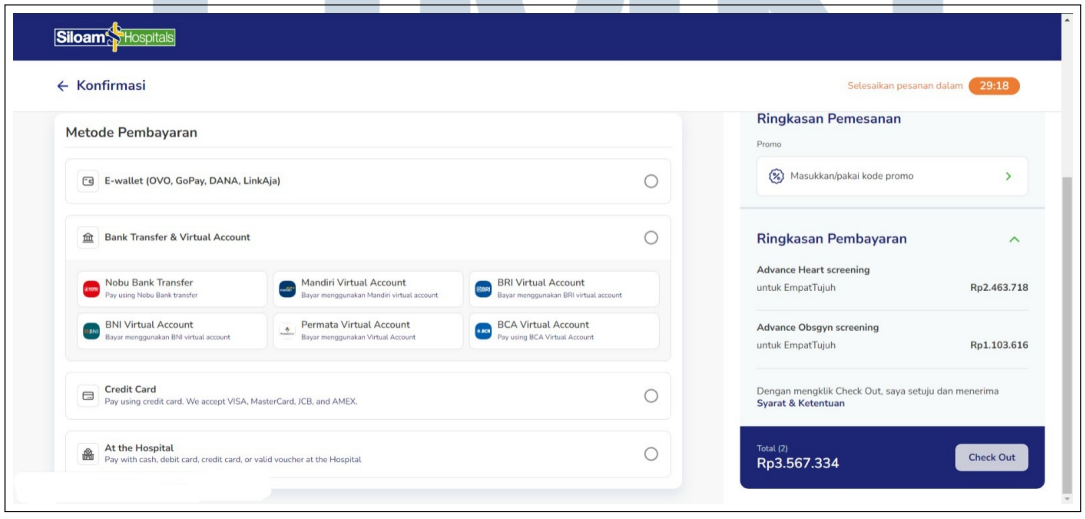
Gambar 3.10. Tampilan Buat Reservasi Paket

Berikut pada Gambar 3.10 adalah halaman reservasi paket yang telah ditambahkan ke keranjang oleh *user*. Sistem *health service* memungkinkan *user* untuk melakukan reservasi lebih dari satu paket. Sistem tidak mengirimkan 2 *event* untuk masing-masing paket, tetapi sistem akan mengirimkan 1 *event* dengan *properties* packages dengan bentuk array dimana packages tersebut menyimpan 2 paket. Namun jika terdapat perbedaan susunan atau perbedaan nama *properties* saat mengirimkan data, *Kibana* hanya dapat memanggil satu field untuk dijadikan *column* pada visualisasi. Jadi, *developer* harus membuat *field* baru untuk menyimpan kedua value tersebut.



Gambar 3.11. Tampilan *Field* Baru

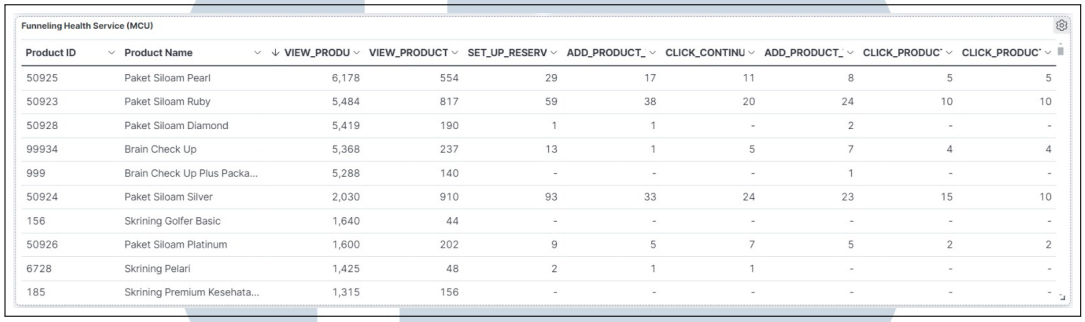
Pada Gambar 3.11, *field* baru yang dibuat menggabungkan *properties* yang hanya mempunyai 1 item_name dan yang dapat mempunyai lebih dari 1 item_name. Dengan digunakannya *field* yang baru kita buat, maka kita dapat membuat *funneling health service* secara keseluruhan.



Gambar 3.12. Tampilan pilih metode pembayaran

Pada Gambar 3.12 adalah tampilan *user* untuk memilih metode pembayaran

paket yang telah di reservasi. Sebagai contoh jika *user* memilih membayar menggunaakn *BCA Virtual Account*, *website* akan menampilkan nomor *virtual account* untuk *user* membayar. *User* yang melakukan *checkout* dan suskses membayar akan dikumpulkan datanya oleh sistem dalam sebuah *event*. *User* yang telah berhasil membayar adalah akhir alur *funneling health service*. Berikut adalah tampilan visualisasi dari *funneling health service*.



Funneling Health Service (MCU)

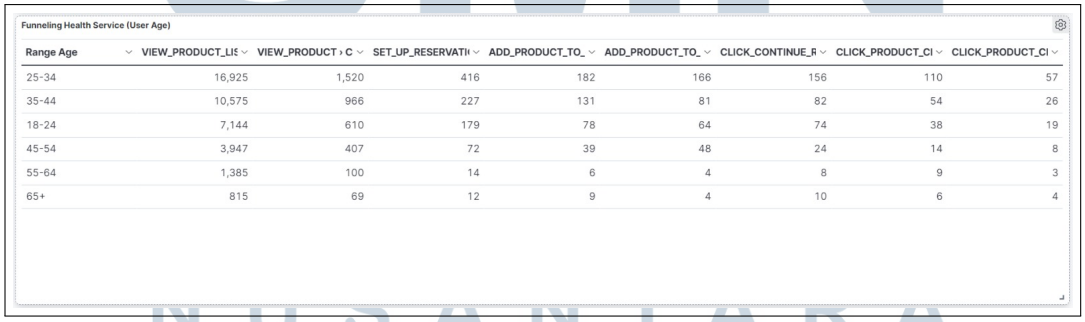
Product ID	Product Name	VIEW_PRODU	VIEW_PRODUCT	SET_UP_RESERV	ADD_PRODUCT_	CLICK_CONTINU	ADD_PRODUCT_	CLICK_PRODUC	CLICK_PRODUC
50925	Paket Siloam Pearl	6,178	554	29	17	11	8	5	5
50923	Paket Siloam Ruby	5,484	817	59	38	20	24	10	10
50928	Paket Siloam Diamond	5,419	190	1	1	-	2	-	-
99934	Brain Check Up	5,368	237	13	1	5	7	4	4
999	Brain Check Up Plus Packa...	5,288	140	-	-	-	1	-	-
50924	Paket Siloam Silver	2,030	910	93	33	24	23	15	10
156	Skrining Golfer Basic	1,640	44	-	-	-	-	-	-
50926	Paket Siloam Platinum	1,600	202	9	5	7	5	2	2
6728	Skrining Pelari	1,425	48	2	1	1	-	-	-
185	Skrining Premium Kesehata...	1,315	156	-	-	-	-	-	-

Gambar 3.13. Visualisasi *Funneling Health Service*

Dari hasil visualisasi tersebut, dapat kita simpulkan bahwa Paket Siloam Pearl adalah paket yang paling banyak dilihat. Untuk paket yang paling banyak berhasil terjual adalah Paket Siloam Ruby dan Paket Siloam Silver.

3.5.2 *Funneling Health Service (User Age)*

Pada visualisasi ini, perjalanan *user* sama seperti diatas. Namun, yang membedakan adalah pada visualisasi *funneling* ini menggunakan *range* umur *user*. Pembuatan visualisasi juga tidak membutuhkan adanya inisiasi *field* baru karena dari awal hingga akhir *funneling*, *properties* yang digunakan masih sama.



Funneling Health Service (User Age)

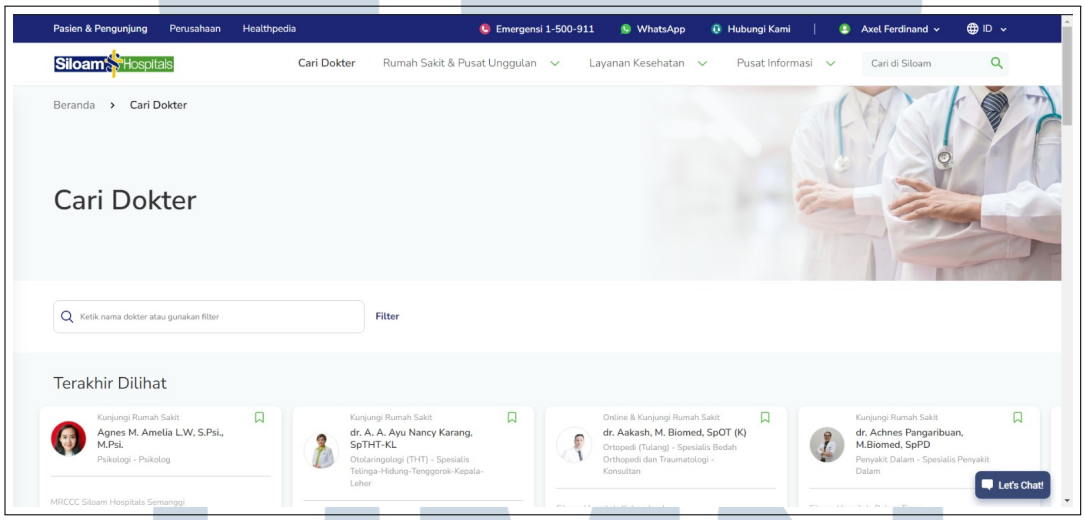
Range Age	VIEW_PRODUCT_LIS	VIEW_PRODUCT › C	SET_UP_RESERVATI	ADD_PRODUCT_TO_	ADD_PRODUCT_TO_	CLICK_CONTINUE_R	CLICK_PRODUCT_CI	CLICK_PRODUCT_CI
25-34	16,925	1,520	416	182	166	156	110	57
35-44	10,575	966	227	131	81	82	54	26
18-24	7,144	610	179	78	64	74	38	19
45-54	3,947	407	72	39	48	24	14	8
55-64	1,385	100	14	6	4	8	9	3
65+	815	69	12	9	4	10	6	4

Gambar 3.14. Visualisasi *Funneling Health Service (User Age)*

Dari hasil visualisasi tersebut, dapat kita lihat bahwa *user* dengan *range* umur 25 - 34 adalah yang paling banyak menggunakan layanan *health service* di

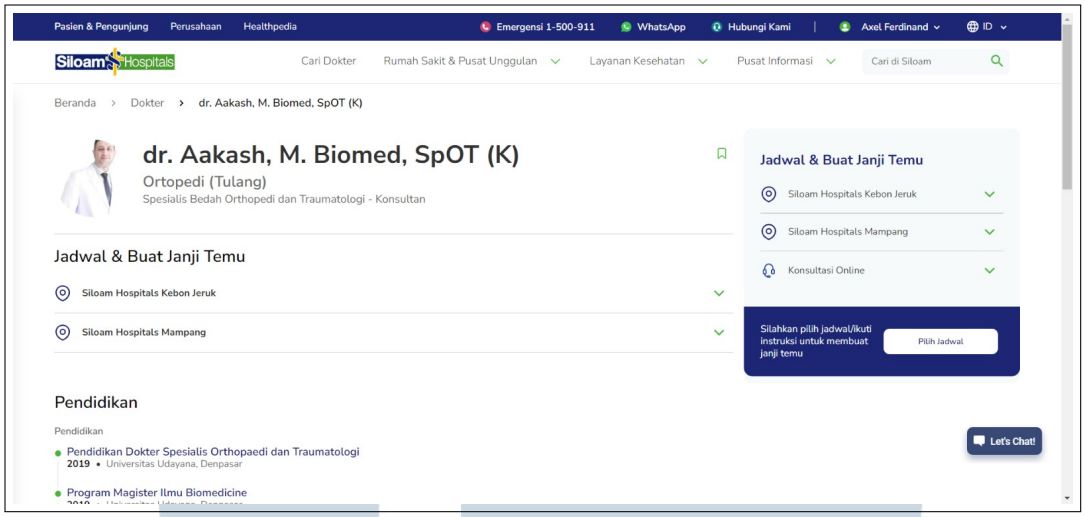
website sedangkan *user* dengan *range* umur 65+ adalah yang paling sedikit. Namun, jika kita lihat bahwa total *records* pada visualisasi diatas terutama pada *event* ”VIEW_PRODUCT_LIST” dan ”VIEW_PRODUCT” lebih banyak daripada yang ada pada visualisasi ini. Hal tersebut disebabkan oleh untuk mendapatkan *range* umur *user*, kita memerlukan *user* untuk login atau membuat akun terlebih dahulu. Pada visualisasi diatas, *records* yang ditampilkan adalah nama paket tersebut dan untuk mendapatkan *value* tidak memerlukan *user* untuk login atau register.

3.5.3 *Funneling Doctor*



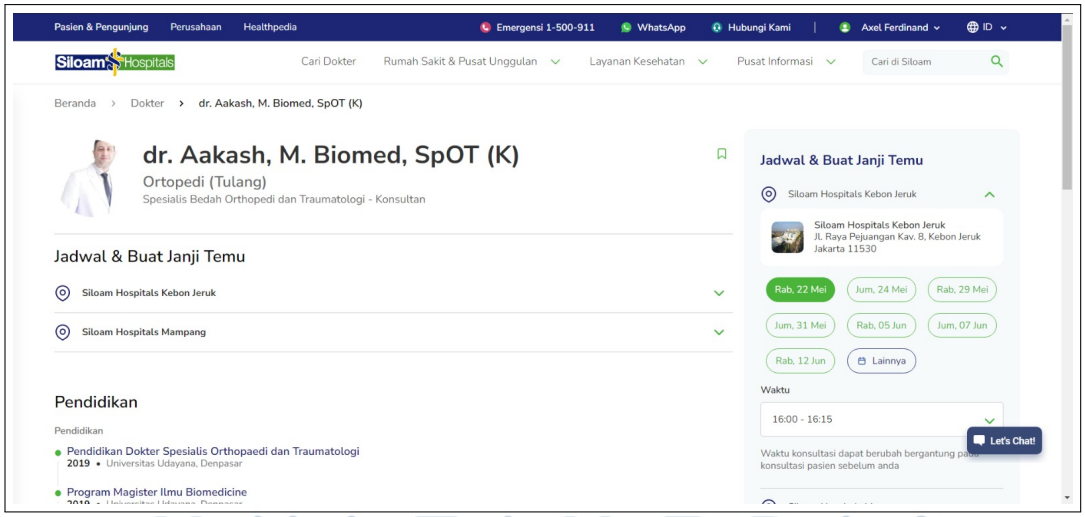
Gambar 3.15. Tampilan melihat semua dokter

Pada visualisasi *funneling doctor*, perjalanan *user* diawali dengan melakukan login dahulu dan membuka halaman lihat semua dokter seperti pada Gambar 3.15. Pada tampilan ini, sistem melakukan pengumpulan data nama dokter sebagai *properties* dan dikirim pada sebuah *event* yang dikirim ke *Elastic*. Jadi pada *event* ini, berisi dokter yang ditampilkan pada halaman ini. Tahap selanjutnya adalah *user* memilih salah satu dokter buka halaman lihat detail dokter.



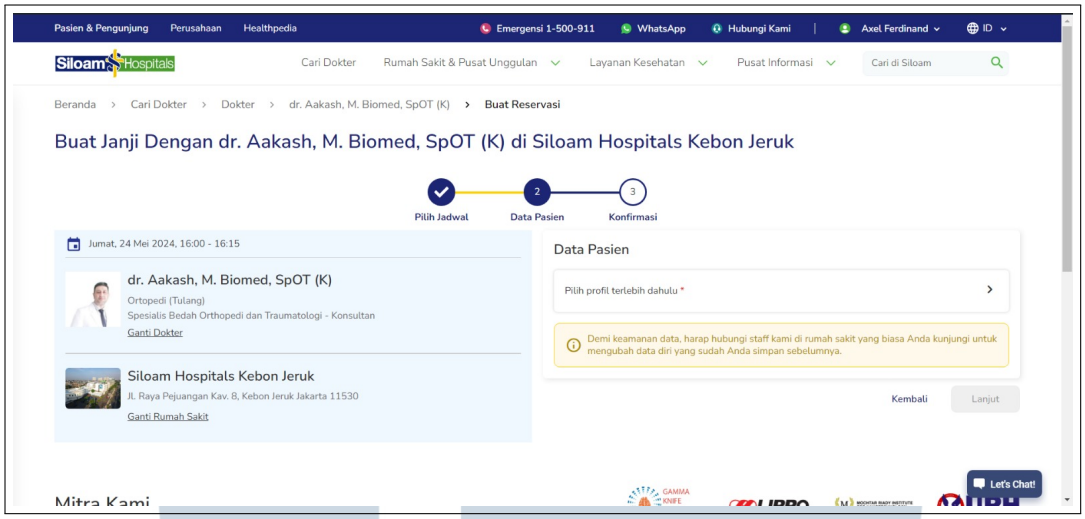
Gambar 3.16. Tampilan melihat detail dokter

Pada Gambar 3.16 adalah tampilan jika *user* membuka halaman detail dokter. Pada saat halaman ini dibuka, sistem akan mengumpulkan nama dokter dalam *proerties* dan mengirimnya sebagai *event*. Tahap selanjutnya, *user* memilih rumah sakit tempat pasien membuat janji temu dengan dokter. Sebagai contoh, *user* memilih untuk membuat janji temu di rumah sakit Siloam Hospitals Kebon Jeruk.

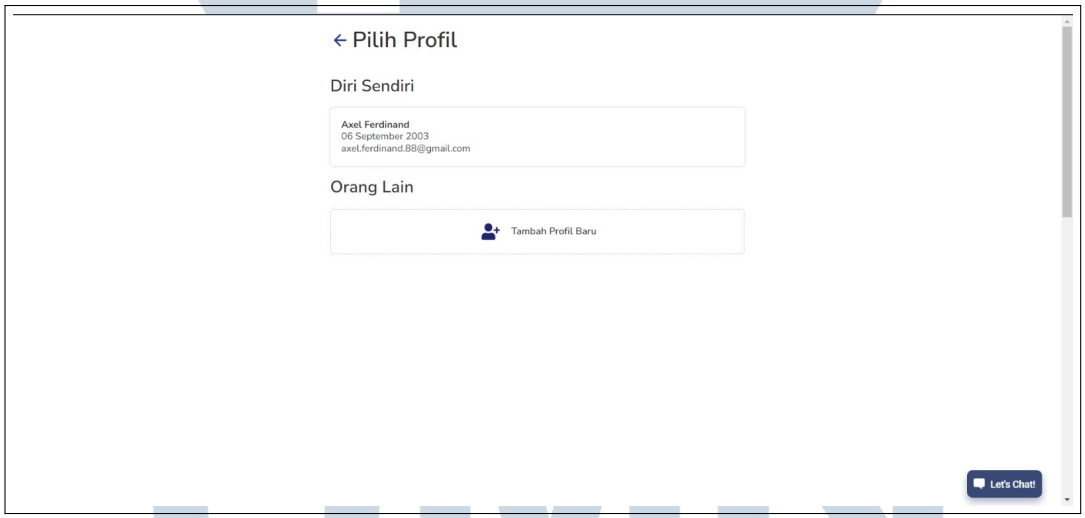


Gambar 3.17. Tampilan memilih tanggal dan waktu pada dokter

Pada Gambar 3.17, adalah tampilan *user* memilih tanggal dan waktu untuk membuat pertemuan dengan dokter. Saat *user* memilih tanggal dan waktu, sistem mengirimkan nama dokter dalam *properties* sebagai *event*. *User* dapat melanjutkan dengan klik *button* pesan di bawah.

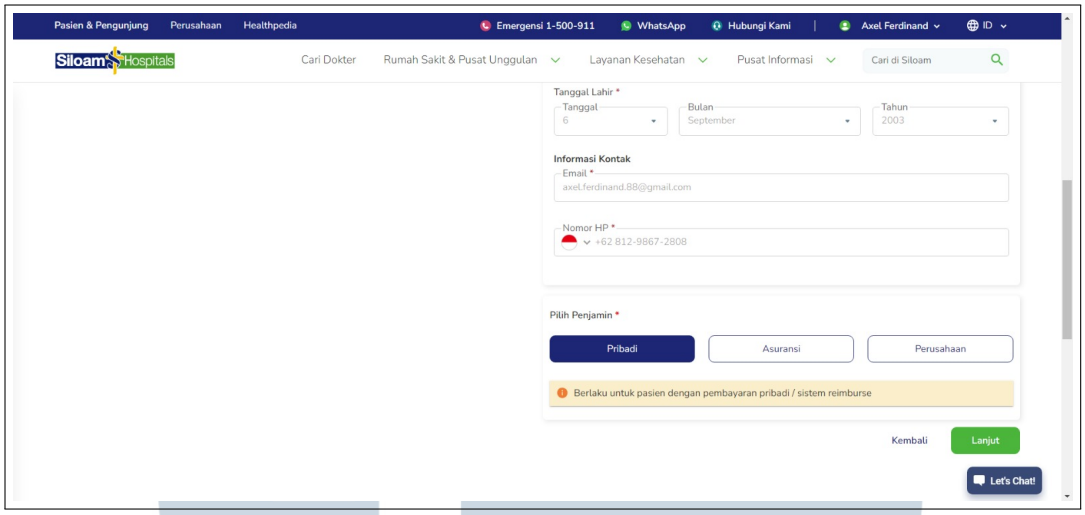


Gambar 3.18. Tampilan membuat reservasi dokter



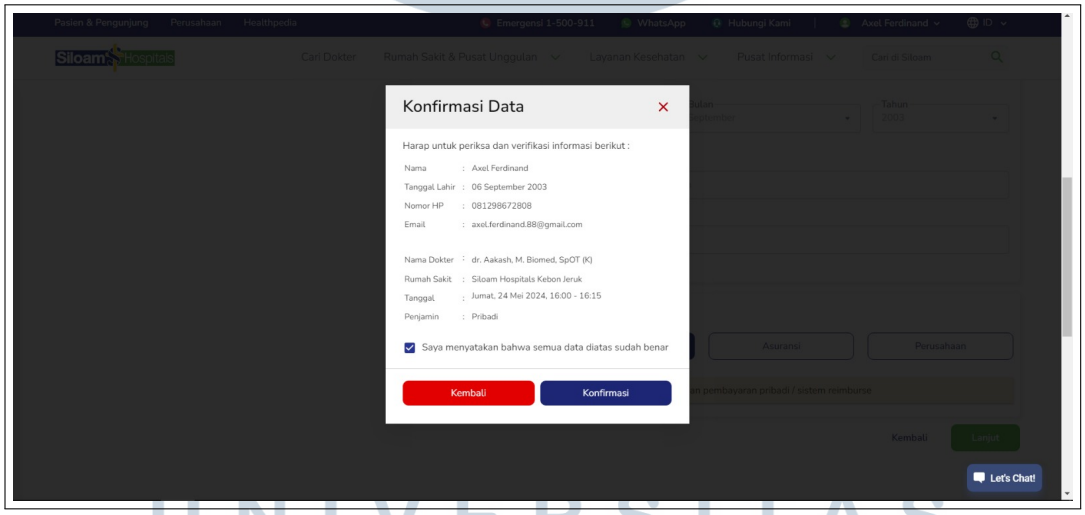
Gambar 3.19. Tampilan memilih pasien

Pada Gambar 3.18, *user* harus memilih pasien mana yang akan melakukan janji temu dengan dokter yang dipilih. Saat *user* memilih pasien, sistem mengirimkan nama dokter dalam *properties* dan dikirim sebagai *event* untuk *funneling*.



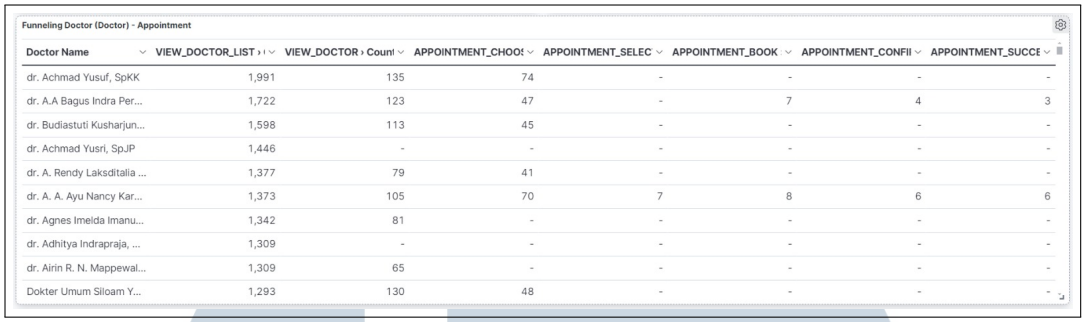
Gambar 3.20. Tampilan memilih penjamin pembayaran

Pada Gambar 3.19 adalah tampilan *user* untuk memilih penjamin pembayaran. Jika *user* telah memilih penjamin pembayaran, *user* dapat memilih lanjut. Ketika *user* memilih lanjut, sistem akan mengirimkan nama dokter dalam *properties* dan mengirimkannya dalam sebuah *event*.



Gambar 3.21. Tampilan melakukan konfirmasi dalam membuat janji temu

Pada Gambar 3.21, *user* harus melakukan konfirmasi terakhir untuk membuat janji temu dengan dokter tersebut. Saat *user* melakukan konfirmasi, sistem akan mengirimkan nama dokter dalam *properties* dan mengirimkan data sebagai *event*. Konfirmasi ini adalah akhir alur dari *funneling doctor*. Namun, beberapa dokter memiliki tipe *secured booking* sehingga beberapa *funneling* harus membayar dahulu untuk mencapai alur terakhir.



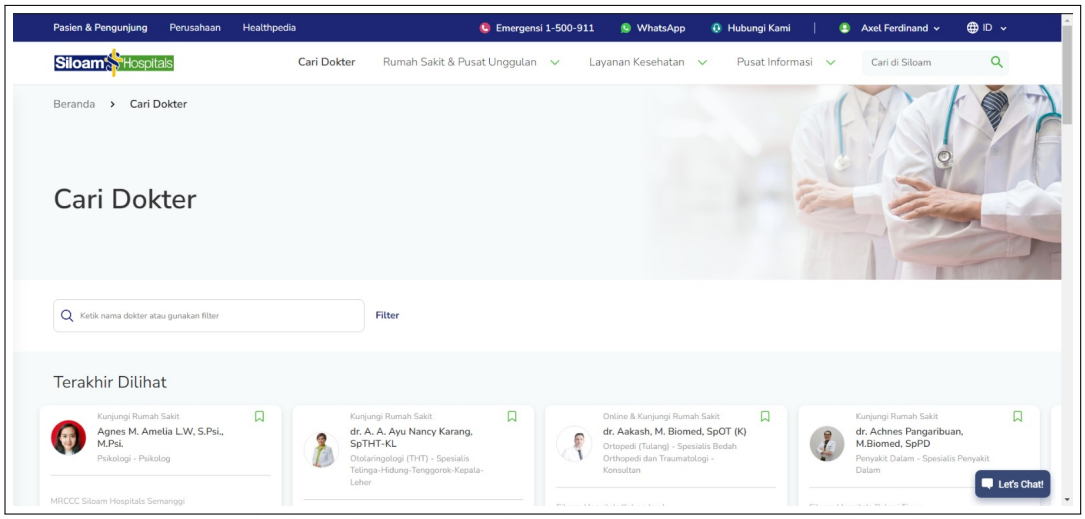
Funneling Doctor (Doctor) - Appointment

Doctor Name	VIEW_DOCTOR_LIST › I	VIEW_DOCTOR › Count	APPOINTMENT_CHOOS	APPOINTMENT_SELEC	APPOINTMENT_BOOK	APPOINTMENT_CONFI	APPOINTMENT_SUCCE
dr. Achmad Yusuf, SpKK	1,991	135	74	-	-	-	-
dr. A.A Bagus Indra Per...	1,722	123	47	-	7	4	3
dr. Budiastuti Kusharjun...	1,598	113	45	-	-	-	-
dr. Achmad Yusri, SpJP	1,446	-	-	-	-	-	-
dr. A. Rendy Laksditalia ...	1,377	79	41	-	-	-	-
dr. A. A. Ayu Nancy Kar...	1,373	105	70	7	8	6	6
dr. Agnes Imelda Imanu...	1,342	81	-	-	-	-	-
dr. Adhitya Indrapraja, ...	1,309	-	-	-	-	-	-
dr. Airin R. N. Mappewal...	1,309	65	-	-	-	-	-
Dokter Umum Siloam Y...	1,293	130	48	-	-	-	-

Gambar 3.22. Visualisasi *Funneling Doctor*

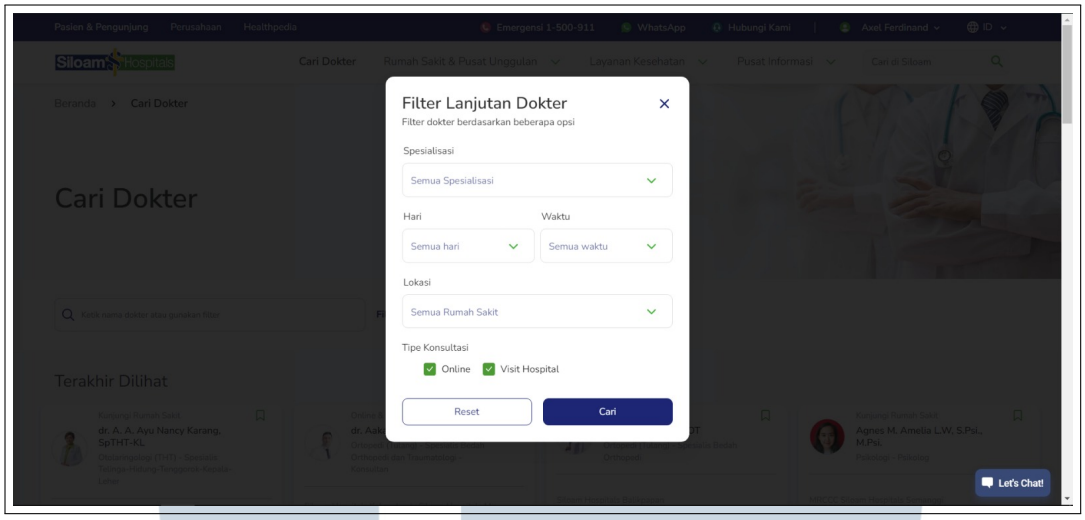
Untuk visualisasi ini tidak membutuhkan inisiasi *field* baru karena setiap *user* hanya dapat membuat janji pertemuan dengan 1 dokter. Jadi, semua *properties* pada setiap *event* memiliki posisi yang sama. Dari hasil visualisasi berikut, dr. Achmad Yusuf, SpKK adalah dokter yang paling banyak dilihat oleh *user*. Dokter yang paling banyak membuat janji temu dengan pasien adalah dr. A. A. Ayu Nancy Karang, SpTHT-KL yaitu dengan total sebanyak 6 kali.

3.5.4 *Rank Search Filter*



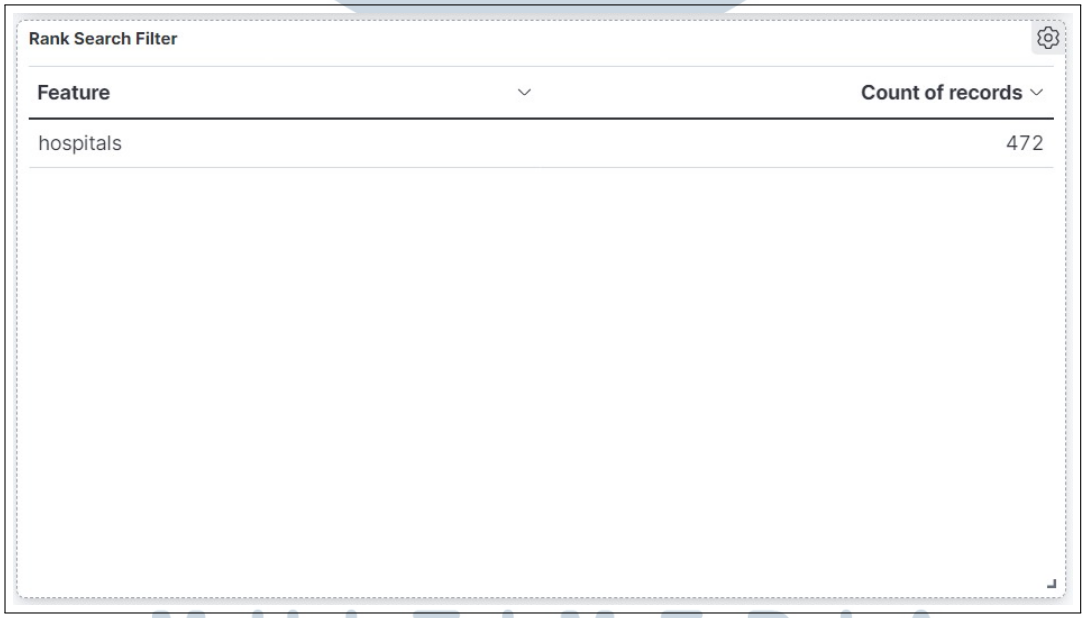
Gambar 3.23. Tampilan melihat semua dokter

Pengumpulan data untuk visualisasi *rank search filter* ada pada halaman daftar dokter. Fitur *search* dapat diakses pada kolom *search bar* pada tengah kiri halaman. Jika *user* mau melakukan *filter*, *user* dapat klik tombol *filter* di sebelah *search bar* untuk membuka modal *filter*.



Gambar 3.24. Tampilan modal *filter* dokter

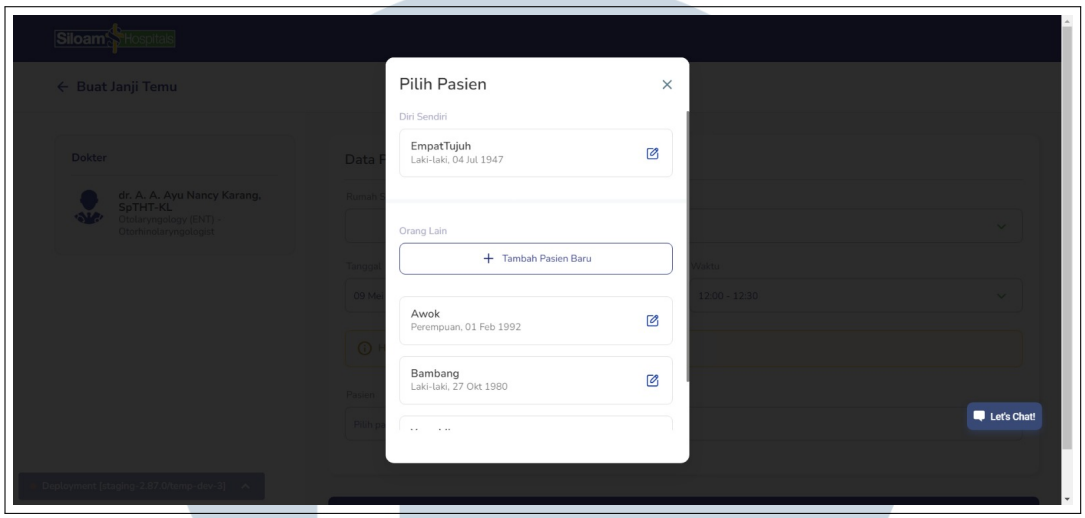
Pada visualisasi *rank filter search*, sistem mengumpulkan data *filter* apa saja yang digunakan oleh *user* saat mencari dokter. Sistem akan mengirim tipe *filter* dan *value* yang *user* gunakan dan dikirim ke *Elastic* sebagai *event*.



Gambar 3.25. Visualisasi *Rank Search Filter*

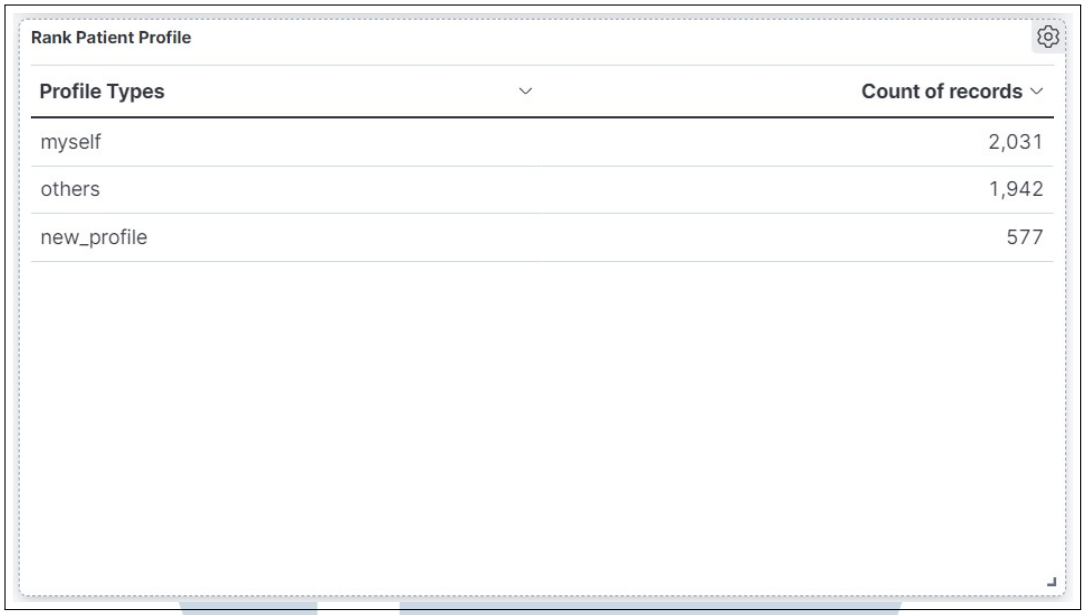
Dari hasil visualisasi tersebut, didapatkan bahwa *filter* mencari dokter berdasarkan rumah sakit adalah *filter* yang paling sering digunakan oleh *user*. Tercatat sebanyak 472 kali *filter* rumah sakit digunakan oleh *user*.

3.5.5 *Rank Patient Profile*



Gambar 3.26. Tampilan pilih pasien pada *funneling doctor*

Pada Gambar 3.26, data tersebut merupakan *fake data* atau *data dummy* yang digunakan selama proses *development*. Sistem akan mengumpulkan data dengan tipe "*myself*" jika *user* memilih akun dengan nama EmpatTujuh karena dia memilih data *user* yang sedang login. Selain itu, jika *user* memilih akun dengan nama Awok atau Bambang, maka sistem akan mengumpulkan data dengan tipe "*others*". Terakhir sistem akan mengumpulkan data dengan tipe "*new_profile*" jika *user* membuat pasien baru.



Gambar 3.27. Visualisasi *Rank Patient Profile*

Dari hasil visualisasi tersebut, didapatkan bahwa *user* paling banyak memilih data akun yang digunakan sebagai pasien. *User* paling sedikit yang membuat data pasien baru saat membuat janji temu dengan dokter.

3.6 Kendala dan Solusi yang Ditemukan

Kendala pada proses pelaksanaan magang adalah seiringnya kita sedang mengerjakan pembuatan *tracker* untuk mengumpulkan data, keinginan dari tim *product* yang menambah atau mengubah *requirement* yang ada. Hal tersebut membuat kita harus mengubah atau bahkan menghapus *tracker* yang sudah dibuat karena perubahan permintaan dari tim *product*.

Solusi dari kendala tersebut adalah dengan menggunakan *management tools* agar *requirement* yang ditambah atau diubah dapat langsung diketahui dari pihak yang lain. Selain itu juga, project dapat dibuat kontrak untuk tidak menambah atau mengubah *requirement* yang sudah ada. Tentunya tim *product* diberikan waktu lebih agar permintaan lebih maksimal.