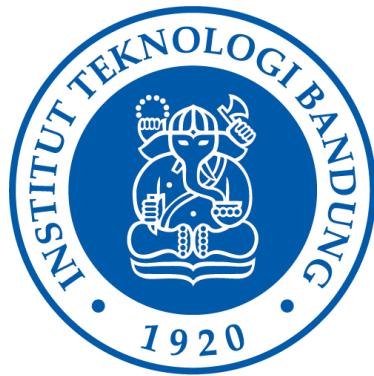


LAPORAN TUGAS BESAR
MATA KULIAH IF1210 ALGORITMA DAN PEMROGRAMAN 1
PROGRAM RUMAH SAKIT NIMON

Dosen Pengampu : Dr. Riza Satria Perdana, S. T, M.T.

Kelas / Kelompok : 03 / A



Disusun oleh:

Leticia Aldina Trulykinanti (18223108)
Laras Hati Mahendra (18223118)
Rezky M. Hafiz Batubara (18224033)
Ellaine Juvina (18224071)
Muhammad Afif H. (18224099)

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

2025

HALAMAN PERNYATAAN

"Saya menyatakan bahwa saya mengerjakan tugas besar ini dengan sejujur-jujurnya, tanpa menggunakan cara yang tidak dibenarkan. Apabila di kemudian hari diketahui saya mengerjakan tugas besar ini dengan cara yang tidak jujur, saya bersedia mendapatkan konsekuensinya, yaitu mendapatkan nilai E pada mata kuliah IF1210 Algoritma dan Pemrograman 1 Semester 2 2024/2025."

Yang mengeluarkan pernyataan,

Leticia Aldina Trulykinanti [18223108]

Laras Hati Mahendra [18223118]

Rezky Muhammad Hafiz Batubara [18224033]

Ellaine Juvina [18224071]

Muhammad Afif Habiburrahman [18224099]

DAFTAR ISI

LAPORAN TUGAS BESAR.....	1
HALAMAN PERNYATAAN.....	2
DAFTAR ISI.....	3
DAFTAR TABEL.....	4
DAFTAR GAMBAR.....	5
A. DESKRIPSI persoalan.....	7
B. RENCANA IMPLEMENTASI.....	8
C. DAFTAR PEMBAGIAN KERJA KELOMPOK.....	9
D. CHECKLIST HASIL PENGERJAAN TUGAS BESAR.....	13
E. DESAIN PERINTAH.....	14
F. DESAIN KAMUS DATA.....	39
G. DESAIN DEKOMPOSISI ALGORITMIK DAN FUNGSIONAL PROGRAM.....	43
H. SPESIFIKASI FUNGSIONAL PROGRAM.....	58
I. LAMPIRAN SOURCE CODE.....	107
II. LAMPIRAN HASIL PENGUJIAN PROGRAM.....	128
III. LAMPIRAN ASISTENSI.....	143

DAFTAR TABEL

Tabel 1 Daftar ADT.....	8
Tabel 2 Daftar Pembagian Kerja Kelompok.....	9
Tabel 3 Daftar Pembagian Pembuatan Laporan.....	12
Tabel 4 Checklist Hasil Pengerjaan Tugas Besar.....	13

DAFTAR GAMBAR

Gambar 1 Nimens	7
Gambar 2 Flowchart login	42
Gambar 3 Register	43
Gambar 4 Flowchart Logout	44
Gambar 5 Flowchart Lupa Password	44
Gambar 6 Flowchart Menu & Help	45
Gambar 7 Flowchart Denah Rumah Sakit	46
Gambar 8 Flowchart Lihat User	48
Gambar 9 Flowchart Cari User	48
Gambar 10 Flowchart Lihat Antrian	49
Gambar 11 Flowchart Tambah Dokter	50
Gambar 12 Flowchart Diagnosis	51
Gambar 13 Flowchart Ngobatin	51
Gambar 14 Flowchart Aku boleh pulang ga, dok 😊?	52
Gambar 15 Flowchart Daftar Check-Up	52
Gambar 16 Flowchart Antrian Saya!	53
Gambar 17 Flowchart Minum Obat	53
Gambar 18 Flowchart Minum Penawar	54
Gambar 19 Flowchart Exit	54
Gambar 20 Flowchart Skip_Antrian	55
Gambar 21 Source code struktur data user.h	104
Gambar 22 Source code header ADT list_user.h	105
Gambar 23 Source code implementasi ADT list_user.h	107
Gambar 25 Source code implementasi ADT set.h	109
Gambar 26 Source code F01: login.c	110
Gambar 27 Source code F02: register.c	111
Gambar 28 Source code F03: logout.c	112
Gambar 29 Source code F04: lupa_password.c	112
Gambar 30 Source code F05: help.c	113
Gambar 31 Source code F06: lihat_denah.c	113
Gambar 32 Source code F10: f10.c	114
Gambar 33 Source code F18: exit.c	115
Gambar 34 Source code Diagnosis	116
Gambar 35 Source code Ngobatin	117
Gambar 36 Source code Daftar Check Up	117
Gambar 37 Source code Antrian Saya!	119
Gambar 38 Source code Struktur Data Pasien	119
Gambar 44 Hasil pengujian F01-Login	125

Gambar 45 Hasil pengujian F02-Register	126
Gambar 46 Hasil pengujian F03-Logout	127
Gambar 47 Hasil pengujian F04-Lupa Password	127
Gambar 48 Hasil pengujian F05-menu dan help	128
Gambar 49 Hasil pengujian F06-Lihat Denah & Ruangan	129
Gambar 50 Hasil pengujian F07- Lihat User	131
Gambar 51 Hasil pengujian F08- Cari user	133
Gambar 52 Hasil pengujian F09- Lihat Antrian	133
Gambar 53 Hasil pengujian F10	134
Gambar 54 Hasil pengujian F11 - Diagnosis	134
Gambar 55 Hasil pengujian F12 - Ngobatin	135
Gambar 56 Hasil pengujian F13	135
Gambar 57 Hasil pengujian F14 - Daftar Check-Up	136
Gambar 58 Hasil pengujian F15 - Antrian Saya!	137
Gambar 59 Hasil pengujian F16	138
Gambar 60 Hasil pengujian F17	138
Gambar 61 Hasil pengujian F18	139
Gambar 62 Hasil pengujian B05	139
Gambar 63 Hasil pengujian B06	139
Gambar 64 Form asistensi pertama	141
Gambar 65 Form asistensi pertama	141
Gambar 67 Form asistensi kedua	143

A. DESKRIPSI PERSOALAN



Gambar 1 Nimons

Tugas Besar ini merupakan tugas untuk membuat sebuah program simulasi sistem informasi rumah sakit yang melibatkan sejumlah tokoh, yaitu pasien, dokter, dan manajer rumah sakit sebagai aktor utama dalam sistem (plot story). Program ini bertujuan untuk menangani berbagai aktivitas dalam lingkungan rumah sakit seperti registrasi pasien, login, logout, pengelolaan data pengguna, dan aktivitas medis lainnya.

Terdapat 16 fungsi wajib yang harus diimplementasikan, mencakup operasi dasar seperti registrasi, login, melihat profil, pencatatan medis, serta manajemen pengguna dan data. Selain itu, tersedia pula 5 fungsi bonus yang dapat dikerjakan untuk menambah nilai, seperti fitur pencarian lanjutan, statistik, atau fitur keamanan tambahan seperti enkripsi password. Dalam proses pengembangan program ini, kami menggunakan bahasa pemrograman C dan memanfaatkan file CSV sebagai basis penyimpanan data permanen.

Namun, terdapat beberapa batasan dalam pembuatan program ini. Kami tidak diperkenankan menggunakan library eksternal dalam bentuk apa pun (selain library standar C seperti stdio.h, string.h, dll), tidak diperbolehkan membuat file CSV sementara (temporary), serta diharuskan mematuhi struktur modular sesuai dengan pembagian file header (.h) dan source (.c). Semua fungsi wajib harus diimplementasikan sendiri tanpa meng

B. RENCANA IMPLEMENTASI

Tabel 1 Daftar ADT

ADT	Fitur Terkait	Deskripsi Implementasi	Alasan Penggunaan
ADT List (Array of Struct)	F01 - Login, F02 - Register Pasien, F07 - Lihat User, F08 - Cari User, F10 - Tambah Dokter	Menyimpan seluruh user (dokter, pasien, manager) dalam array statis yang ukurannya dibatasi.	Akses cepat dengan indeks dan sederhana untuk pencarian berurutan dan sorting.
ADT Set	F02 - Register Pasien, F10 - Tambah Dokter	Menyimpan username dalam struktur set (tanpa duplikat, case-insensitive).	Untuk validasi keunikan username yang sensitif terhadap duplikasi (case-insensitive).
ADT Stack	F16 - Minum Obat, F17 - Minum Penawar, F13 - Pulang Dok	Stack digunakan untuk menyimpan urutan obat yang diminum oleh pasien, karena urutan konsumsi penting.	Stack cocok untuk model LIFO (Last-In First-Out), di mana obat terakhir yang diminum bisa dikeluarkan lebih dulu (F17).
ADT Queue	F06 - Denah Rumah Sakit (ruangan), F09 - Lihat Antrian, F13 - Pulang Dok, F14 - Daftar Check-up	Queue digunakan untuk mengatur antrian pasien yang mendaftar ke dokter secara FIFO.	Mengatur alur pasien secara adil berdasarkan urutan pendaftaran.
ADT Linked List	F14 - Daftar Check-up, F15 - Antrian Saya!, B06 - Mainan Antrian	Mewakili antrian per ruangan dokter, terhubung antar node pasien.	Digunakan karena lebih fleksibel dibanding array, terutama jika jumlah pasien dinamis.
ADT Map	F11 - Diagnosis, F12 - Ngobatin	Menyimpan relasi penyakit → daftar obat dan pasien → kondisi checkup.	Efisien untuk mapping penyakit ke obat dan lookup data pasien.

C. DAFTAR PEMBAGIAN KERJA KELOMPOK

Tabel 2 Daftar Pembagian Kerja Kelompok

Fitur	Implementasi	NIM Desainer	NIM Coder	NIM Tester
F00 - Rencana Implementasi	ADT List, ADT Set, ADT Stack, ADT Queue, ADT Linked List, ADT Map	18224099	18224099	18224099
F01 - Login	Dibuat prosedur login_system untuk memverifikasi username dan password, serta menetapkan current_user sesuai data yang valid di sistem.	18223118	18223118 18224099	18223118 18224099
F02 - Register Pasien	Membuat fungsi pendukung is_username_unique yang berfungsi untuk melakukan pemeriksaan keunikan username dan membuat prosedur register_pasien yang akan menghubungkan proses input data dari pengguna.	18223118	18223118	18223118 18224099
F03 - Logout	Membuat prosedur logout_system yang akan menghubungkan kondisi status login pengguna dengan prosedur keluar dari akun secara sistematis.	18223118	18223118	18223118 18224099
F04 - Lupa Password	Membuat prosedur lupa_password_system yang akan menghubungkan proses verifikasi identitas pengguna dengan proses mereset password dan prosedur generate_kode_unik	18223118	18223118	18223118 18224099
F05 - Menu & Help	Dibuat prosedur 'help_system' untuk menampilkan daftar command sesuai peran pengguna. Jika belum login, hanya command dasar seperti	18223118	18223118	18223118 18224099

	login dan register yang ditampilkan.			
F06 - Denah Rumah Sakit	Membuat prosedur lihatdenah untuk menampilkan daftar ruangan dalam bentuk tabel, dan lihat_ruangan untuk menampilkan detail isi ruangan tertentu sesuai nomor yang dipilih.	18224033	18224033	18224033 18224099
F07 - Lihat User	Membuat prosedur lihat_user, lihat_pasien, dan lihat_dokter untuk menampilkan semua user ataupun pasien dan dokter secara spesifik berdasarkan id/nama dan ascending/descending.	18224033	18224033	18224033
F08 - Cari User	Membuat prosedur cari_user, cari_pasien, dan cari_dokter berdasarkan id dan nama, serta penyakit khusus untuk pasien, yang akan menampilkan user, pasien, maupun dokter yang sesuai dengan id, nama, maupun penyakit yang sesuai.	18224033	18224033	18224033
F09 - Lihat Antrian	Membuat prosedur lihat_semua_antrian yang dapat menampilkan semua ruangan dengan dokter di dalamnya, nomor ruangan, kapasitas ruangan, serta pasien di ruangan dan pasien di antrian jika ada.	18224099	18224099	18224099
F10 - Tambah Dokter	Membuat prosedur tambah_dokter untuk menambahkan akun dokter baru oleh Manager, serta prosedur assign_dokter untuk menetapkan dokter ke ruangan tertentu.	18224033	18224033	18224033 18224099
F11 - Diagnosis	Membuat fungsi identifikasi_penyakit untuk menentukan penyakit pasien berdasarkan data medis dan prosedur diagnosis_pasien bagi dokter untuk mendiagnosa pasien.	18224071	18224071	18224071 18224099

F12 - Ngobatin	Membuat fungsi cari_obat untuk mencari obat yang sesuai dengan nama penyakit pasien dan prosedur ngobatin bagi dokter untuk memberikan resep obat kepada pasien.	18224071	18224071	18224071 18224099
F13 - Aku boleh pulang ga dok?	Membuat prosedur make_default_pasien untuk mereset semua data pasien dan prosedur pulangdok untuk mengecek status pasien dan apakah sudah memenuhi syarat untuk diperbolehkan pulang atau tidak	18223108	18223108	18223108
F14 - Daftar Check-up	Membuat fungsi validasi_float dan validasi_integer untuk memastikan input data medis valid, prosedur display_dokter untuk menampilkan daftar dokter yang tersedia, dan prosedur daftar_check_up bagi pasien untuk mendaftar check-up.	18224071	18224071	18224071 18224099
F15 - Antrian Saya	Membuat prosedur cek_antrian_saya untuk menampilkan status antrian pasien.	18224071	18224071	18224071 18224099
F16 - Minum Obat	Membuat prosedur minum_obat yang dapat memperbolehkan pasien untuk meminum obat yang sudah diresepkan oleh dokter.	18223108	18223108	18223108
F17 - Minum Penawar	Membuat prosedur penawar yang dapat memperbolehkan pasien untuk meminum obat penawar apabila mengalami implikasi obat (urutan obat yang diminum salah) sehingga menyebabkan sakit.	18223108	18223108	18223108
F18 - Exit	Dibuat prosedur exit_system yang menanyakan konfirmasi pengguna untuk keluar dari sistem dan menghentikan program jika disetujui.	18223108	18223108	18223108 18224099

B05 - Dead or Alive	Membuat sistem nyawa bagi pasien yang berkurang apabila terjadi kesalahan dalam urutan minum obat	18224099, 18224071	18224099	18224099
B06 - Mainin Antrian	Membuat prosedur skip_antrian yang memungkinkan pasien mengubah posisi antrian menjadi terdepan	18224099, 18224071	18224099	18224099

Tabel 3 Daftar Pembagian Pembuatan Laporan

No.	Bagian Laporan	NIM
1	Halaman Cover	18223108
2	Daftar Isi	18223108, 18223118
3	Daftar Tabel	18223108, 18223118
4	Daftar Gambar	18223108, 18223118
5	Deskripsi Persoalan	18223118
6	Daftar Pembagian Tugas	18223108, 18223118, 18224033, 18224071, 18224099
7	Checklist Hasil Rangkaian, Implementasi, dan Uji Coba	18223108, 18223118, 18224033, 18224099, 18224071
8	Desain Perintah	18223108, 18223118, 18224033, 18224071
9	Desain Kamus Data	18223108, 18223118, 18224099, 18224033, 18224071
10	Desain Dekomposisi Algoritmik dan Fungsional Program	18224099, 18223118, 18223108, 18224033, 18224071
11	Spesifikasi	18223108, 18223118, 18224033, 18224071, 18224099
12	Hasil Pengujian Program	18223118, 18224099
13	Catatan dan Lampiran	18223118, 18224099, 18224071, 18224033

D. CHECKLIST HASIL PENGERJAAN TUGAS BESAR

Tabel 4 Checklist Hasil Pengerjaan Tugas Besar

Fitur	Desain	Implementasi	Testing
F01 - Register	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F02 - Login	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F03 - Logout	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F04 - Lupa Password	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F05 - Menu & Help	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F06 - Denah Rumah Sakit	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F07 - Lihat User	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F08 - Cari User	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F09 - Lihat Antrian	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F10 - Tambah Dokter	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F11 - Diagnosis	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F12 - Ngobatin	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F13 - Aku boleh pulang ga, dok 😊?	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F14 - Daftar Check-Up	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F15 - Antrian Saya!	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
F16 - Minum Obat	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
F17 - Minum Penawar	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
F18 - Exit	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
B05 - Dead or Alive	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
B06 - Mainin Antrian	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

E. DESAIN PERINTAH

1. F01 - Login

```
PROCEDURE login_system()
{ IS: Tidak ada user yang login. User memasukkan username dan password. }
{ FS: Jika username dan password cocok dengan data sistem, user berhasil login dan diberikan sambutan berdasarkan peran.
    Jika username tidak terdaftar, sistem menampilkan pesan bahwa user tidak terdaftar sebagai Manager, Dokter, atau Pasien
    Jika password salah, sistem menampilkan pesan kegagalan login karena kesalahan password dan akan kembali tanpa melanjutkan login. }

# User berhasil login dengan username nimonsslatte dan password yang sesuai
# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja

# Kasus 1: Login sebagai Manager
# Masukkan
>>> LOGIN
Username: nimonsslatte
Password: nimonatutgajah23

# Sistem berhasil menyambut user dengan ucapan selamat pagi menggunakan username nimonsslatte yang berhasil login.
Selamat pagi Manager nimonsslatte!

# Keluaran
# User diberikan deskripsi dengan menggunakan perintah "help" untuk melihat daftar perintah yang tertera.

# User berhasil login dengan username Neroifa dan password yang sesuai

# Kasus 2: Login sebagai Dokter
# Masukkan
>>> LOGIN
Username: Neroifa
Password: Neroifa123

# Sistem berhasil menyambut user dengan ucapan selamat pagi menggunakan username Neroifa yang berhasil login.
Selamat pagi Dokter Neroifa!

# Keluaran
# User diberikan deskripsi dengan menggunakan perintah "help" untuk melihat daftar perintah yang tertera.

# User berhasil login dengan username GRO dan password yang sesuai

# Kasus 3: Login sebagai Pasien
# Masukkan
>>> LOGIN
Username: GRO
Password: NeroifaCantik

# Sistem berhasil menyambut user dengan ucapan selamat pagi menggunakan username GRO yang berhasil login.
Selamat pagi GRO! Ada keluhan apa ?

# Keluaran
```

```

# User diberikan deskripsi dengan menggunakan perintah “help” untuk melihat daftar
perintah yang tertera.

# User gagal login dengan username NONEXISTENT_USER

# Kasus 4: Tidak ada username yang terdaftar
# Masukkan
>>> LOGIN
Masukan username: NONEXISTENT_USER
Masukan password: passwordrandom

# Keluaran
# User diberikan deskripsi mengenai pesan bahwa log in gagal karena user sudah log
in dengan username tersebut sebab tidak ada manager yang bernama NONEXISTENT_USE.
Tidak ada Manager, Dokter, atau pun Pasien yang bernama NONEXISTENT_USER!

```



```

# User gagal login dengan username nimonsganteng

# Kasus 5: Kasus password salah
# Masukkan
>>> LOGIN
Masukan username: nimonsganteng
Masukan password: wrongpassword

# Keluaran
# User diberikan deskripsi mengenai pesan bahwa log in gagal karena user sudah log
in dengan username tersebut sebab password salah.
Password salah untuk pengguna yang bernama nimonsganteng!

```

2. F02 - Register Pasien

```

PROCEDURE register_pasien()
{ IS: Program dalam keadaan aktif dan tidak ada user yang sedang login. User ingin melakukan
pendaftaran sebagai pasien dengan memasukkan username dan password. }
{ FS: Jika username valid dan belum digunakan serta password benar, maka pasien ditambahkan dan
sistem menampilkan pesan berhasil.


- Jika ada user yang sedang login, maka registrasi dibatalkan dan ditampilkan peringatan untuk
logout terlebih dahulu.
- Jika username mengandung karakter non-alfabet, maka registrasi dibatalkan dan sistem
menampilkan pesan kesalahan.
- Jika username sudah terdaftar, maka registrasi dibatalkan dan ditampilkan pesan bahwa nama
sudah digunakan.
- Jika input tidak valid atau kapasitas penuh, maka registrasi tidak dilanjutkan dan sistem
menampilkan pesan yang sesuai}

}

# Kasus 1: Pasien bernama Denis belum ada
# User melakukan registrasi dengan username “Denis” yang belum terdaftar sebelumnya.
# Masukkan
>>> REGISTER
Username: Denis
Password: aditgimananihdit

# Sistem berhasil menambahkan pasien baru dengan nama Denis.
Pasien Denis berhasil ditambahkan!

# User kemudian melakukan login menggunakan username dan password yang telah
didafarkan.
# Masukkan
```

```

>>> LOGIN
Username: Denis
Password: aditgimananihdit

# Sistem menyambut user dengan sapaan selamat pagi sesuai peran pasien.
Selamat pagi Denis! Ada keluhan apa ?

# Pasien baru bernama Denis berhasil diregistrasikan dan dapat login dengan username
serta password yang sesuai.
# Sistem menyambut pasien yang berhasil login dan siap menampilkan daftar perintah
dengan perintah "help".

# Kasus 2: Pasien bernama Denis sudah ada.

# Masukkan
>>> REGISTER
Username: Denis
Password: aditgimananihdit

# Sistem menolak registrasi karna nama Denis sudah terdaftar sebagai pasien.
Registrasi gagal! Pasien dengan nama Denis sudah terdaftar.

# Keluaran
# Sistem membatalkan proses registrasi dan menampilkan pesan bahwa nama pasien sudah
digunakan.
# User tidak dapat mendaftarkan akun dengan username yang sama.

```

3. F03 - Logout

```

PROCEDURE logout_system()
{ IS: Program dalam keadaan berjalan, dan variabel current_user menunjuk ke user yang sedang login
(jika ada). }
{ FS: Jika ada user yang sedang login, maka user berhasil logout dan current_user diset menjadi NULL.
Jika belum login, maka sistem menampilkan pesan bahwa logout gagal dan meminta user login
terlebih dahulu. }

# DISCLAIMER: Tampilan/ interface dibebaskan, berikut hanya contoh saja
# Kasus 1: sedang dalam keadaan logged in
# User melakukan logout setelah berhasil login sebelumnya.
# Masukkan
>>> LOGOUT
# Keluar dari akun

Sampai jumpa!

# Keluaran
# User berhasil logout dan sistem menyambut dengan pesan "Sampai jumpa!".
# Setelah logout, user tidak lagi terhubung dengan sistem (current_user diset ke
NULL).

# Kasus 2: sedang dalam keadaan belum logged in
# Masukkan
>>> LOGOUT
Logout gagal!
Anda belum login, silahkan login terlebih dahulu sebelum melakukan logout

# Keluaran

```

```
# Sistem menampilkan pesan bahwa user belum login dan meminta untuk login terlebih dahulu sebelum dapat logout.
```

4. F04 - Lupa Password

PROCEDURE lupa_password()

```
{ IS: Program dalam keadaan berjalan, dan variabel current_user menunjuk ke user yang sedang login (jika ada). }  
{ FS: - Jika username ditemukan dan kode unik yang dimasukkan cocok, maka user akan diminta untuk memasukkan password baru, dan password user berhasil diperbarui.  
- Jika username tidak ditemukan, maka sistem menampilkan pesan bahwa username tidak terdaftar.  
- Jika kode unik yang dimasukkan tidak sesuai dengan kode unik yang dihasilkan sistem, maka sistem membatalkan proses dan menampilkan pesan bahwa kode unik salah.  
- Jika ada input yang tidak valid, sistem membatalkan proses dan menampilkan pesan kesalahan input. }
```

```
# Kasus 1: Manager, Dokter, atau Pasien bernama Jeffreey ada
```

```
# Masukkan  
>>> LUPA_PASSWORD  
Username: Jeffreey  
Kode Unik: Je2fr2ey
```

```
Halo Dokter Jeffreey, silakan daftarkan ulang password anda!  
Password Baru: JR1234
```

```
>>> LOGIN  
Username: Jeffreey  
Password: JR1234
```

```
Selamat pagi Dokter Jeffreey!
```

Keluaran

```
# Sistem berhasil mengenali username, mencocokkan kode unik, dan meminta user untuk memasukkan password baru.  
# Setelah berhasil, user dapat login menggunakan password yang baru.
```

```
# Kasus 2: Manager, Dokter, atau Pasien bernama NONEXISTENT_USER tidak ada
```

```
# Masukkan  
>>> LUPA_PASSWORD  
Username: NONEXISTENT_USER  
Kode Unik: NONEXISTENT_USER
```

```
Username tidak terdaftar!
```

Keluaran

```
# Sistem tidak menemukan username dalam data pengguna dan menampilkan pesan bahwa username tidak terdaftar.
```

```
# Kasus 3: Kode unik untuk username nimonsslatte bukan nimonsslatte tetapi adalah nimon2sla2te
```

```
>>> LUPA_PASSWORD  
Username: nimonsslatte  
Kode Unik: nimonsslatte
```

```
Kode unik salah!
```

Keluaran

```
# Username ditemukan, namun kode unik tidak cocok sehingga proses ubah password dibatalkan dengan pesan kesalahan.
```

5. F05 - Menu & Help

PROCEDURE help_system()

```
{ IS: Program dalam keadaan berjalan. Pengguna dapat belum login atau sudah login sebagai Manager, Dokter, atau Pasien. }
{ FS: Sistem menampilkan daftar perintah yang bisa digunakan sesuai dengan status login dan peran pengguna (jika sudah login).
  - Jika belum login, sistem menampilkan instruksi untuk login atau register. }
```

```
# Kasus 1: belum dalam keadaan logged in
>>> HELP
```

```
===== HELP =====
```

Kamu belum login sebagai role apapun. Silahkan login terlebih dahulu.

1. LOGIN: Masuk ke dalam akun yang sudah terdaftar
2. REGISTER: Membuat akun baru

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

```
# Kasus 2: Sudah login sebagai Dokter
>>> HELP
```

```
===== HELP =====
```

Halo Dokter [Neroifa](#). Kamu memanggil command HELP. Kamu pasti sedang kebingungan. Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

1. LOGOUT: Keluar dari akun yang sedang digunakan
2. DIAGNOSIS: Melakukan diagnosis penyakit pasien berdasarkan kondisi tubuh pasien
3. # ...dan seterusnya

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

```
# Kasus 3: Sudah login sebagai Pasien
>>> HELP
```

```
===== HELP =====
```

Selamat datang, [GRO](#). Kamu memanggil command HELP. Kamu pasti sedang kebingungan. Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

1. LOGOUT: Keluar dari akun yang sedang digunakan
2. DAFTAR_CHECKUP: Mendaftarkan diri untuk pemeriksaan dokter
3. # ...dan seterusnya

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

```
# Kasus 4: Sudah login sebagai Dokter  
">>>> HELP
```

```
===== HELP =====
```

Halo Manager [nimonsslatte](#). Kenapa kamu memanggil command HELP? Kan kamu manager, tapi yasudahlah kamu pasti sedang kebingungan. Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

1. LOGOUT: Keluar dari akun yang sedang digunakan
2. TAMBAH_DOKTER: Mendaftarkan dokter baru ke sistem
3. # ...dan seterusnya

Footnote:

1. Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
2. Jangan lupa untuk memasukkan input yang valid

6. F06 - Denah Rumah Sakit

- Lihat Denah

PROCEDURE lihat_denah (Input ruangan: ListRuang)

{ IS: Program dalam keadaan berjalan. Pengguna dapat belum login atau sudah login sebagai Manager, Dokter, atau Pasien. }
{ FS: Sistem menampilkan daftar perintah yang bisa digunakan sesuai dengan status login dan peran pengguna (jika sudah login).
- Jika belum login, sistem menampilkan instruksi untuk login atau register. }

```
>>> LIHAT_DENAH
```

```
+---+---+---+---+---+---+---+---+  
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  
+---+---+---+---+---+---+---+
```

- Lihat Ruangan

PROCEDURE lihat_ruang (Input/output ruangan: ListRuang, input: Integer, input/output users: ListUser)

{ IS: Program dalam keadaan berjalan. Pengguna dapat belum login atau sudah login sebagai Manager, Dokter, atau Pasien. }
{ FS: Sistem menampilkan daftar perintah yang bisa digunakan sesuai dengan status login dan peran pengguna (jika sudah login).
- Jika belum login, sistem menampilkan instruksi untuk login atau register. }

```
# Kasus 1: ruangan terdapat dokter dan pasien
```

```
>>> LIHAT_RUANGAN 1
```

```
--- Detail Ruangan 1 ---
```

Kapasitas : 3
Dokter : Dr. Strange
Pasien di dalam ruangan:
1. John Smith
2. John Lennon

```
-----
```

```

# Kasus 2: ruangan terdapat dokter dan tidak ada pasien
>>> LIHAT_RUANGAN 2

--- Detail Ruangan 2 ---
Kapasitas : 3
Dokter    : Dr. Oz
Pasien di dalam ruangan:
Tidak ada pasien di dalam ruangan saat ini.
-----

# Kasus 3: ruangan tidak terdapat dokter.
>>> LIHAT_RUANGAN 3

--- Detail Ruangan 3 ---
Kapasitas : 3
Dokter    : -
Pasien di dalam ruangan:
Tidak ada pasien di dalam ruangan saat ini.
-----

```

7. F07 - Lihat User

- To lower

FUNCTION to_lower (input c: char) -> char
{ IS: Sebuah karakter alfabet diberikan. }
{ FS: Jika karakter huruf kapital, dikonversi menjadi huruf kecil.
Jika bukan huruf kapital, dikembalikan apa adanya. }

- Compare nama

FUNCTION compare_nama (input/output a: char, input/output b: char) -> integer
{ IS: Dua string nama diberikan dalam huruf campuran kapital dan kecil. }
{ FS: Mengembalikan -1 jika $a < b$, 0 jika $a == b$, dan 1 jika $a > b$, dibandingkan secara tidak case-sensitive. }

- Sort user

PROCEDURE sort_user (input: L: ListUser, input idx: array of integer, input len: integer, input id: integer, input asc: integer)
{ IS: Array indeks `idx` sudah terisi dengan indeks user yang ingin diurutkan. }
{ FS: Indeks user dalam `idx` diurutkan berdasarkan ID jika `id = 1`, atau berdasarkan nama jika `id = 2 (false/0)`, secara ascending jika `asc = 1`, atau descending jika `asc = 2 (false/0)`. }

- Print user

PROCEDURE print_user (input: L: ListUser, idx: array of integer, len: integer, view: integer)
{ IS: Program dalam keadaan berjalan dan pengguna telah login. Data pengguna sudah tersedia dalam `L` dan sudah disortir dalam `idx`. }
{ FS: Menampilkan daftar user ke layar sesuai mode tampilan `view`: 0 = semua user, 1 = hanya pasien, 2 = hanya dokter. Data manajer tidak ditampilkan. }

- Lihat user

PROCEDURE lihat_user (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna telah login. }

{ FS: Jika pengguna merupakan Manager, sistem akan menampilkan seluruh data pengguna (selain Manager),
 disortir berdasarkan ID atau Nama secara ascending atau descending sesuai pilihan.
 Jika bukan Manager, ditampilkan pesan akses ditolak.
 Jika belum login, ditampilkan pesan bahwa pengguna belum login. }

- Lihat pasien

PROCEDURE lihat_pasien (Input L: ListUser)
 { IS: Program dalam keadaan berjalan. Pengguna telah login. }
 { FS: Jika pengguna merupakan Manager, sistem akan menampilkan seluruh data pasien,
 disortir berdasarkan ID atau Nama secara ascending atau descending sesuai
 pilihan pengguna. Jika bukan Manager, ditampilkan pesan "Akses ditolak".
 Jika belum login, ditampilkan pesan bahwa pengguna belum login. }

- Lihat dokter

PROCEDURE lihat_pasien (Input L: ListUser)
 { IS: Program dalam keadaan berjalan. Pengguna telah login. }
 { FS: Jika pengguna merupakan Manager, sistem akan menampilkan seluruh data dokter,
 disortir berdasarkan ID atau Nama secara ascending atau descending sesuai
 pilihan pengguna. Jika bukan Manager, ditampilkan pesan "Akses ditolak".
 Jika belum login, ditampilkan pesan bahwa pengguna belum login. }

```

# DISCLAIMER: Tampilan/interface dibebaskan, berikut hanya contoh saja
# Kasus 1: Melihat data user (user bisa berarti dokter atau pasien)
>>> LIHAT_USER
Urutkan berdasarkan?
1. ID
2. Nama
>>> Pilihan: 1

```

```

Urutan sort?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 1

```

Menampilkan semua pengguna dengan ID terurut ascending...

ID	Nama	Role	Penyakit
1	Jeffrey	Dokter	-
2	Erik	Pasien	Maag
3	Neroifa	Dokter	-
4	Daev	Pasien	Flu
5	Remon	Pasien	Maag
6	Alpin	Dokter	-

Kasus 2: Spesifik melihat data pasien

```

>>> LIHAT_PASIEN
Urutkan berdasarkan?
1. ID

```

```

2. Nama
>>> Pilihan: 2

Urutan sort?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 2

Menampilkan pasien dengan nama terurut descending...
ID | Nama      | Penyakit
-----
5  | Remon     | Maag
2  | Erik      | Maag
4  | Daev      | Flu

```

```

# Kasus 3: Spesifik melihat data dokter
>>> LIHAT_DOKTER
Urutkan berdasarkan?
1. ID
2. Nama
>>> Pilihan: 2

Urutan sort?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 1

Menampilkan dokter dengan nama terurut ascending...
ID | Nama
-----
6  | Alpin
1  | Jeffrey
3  | Neroifa

```

8. F08 - Cari User

- Binary search

```

FUNCTION binary_search (input: L: ListUser, input: targetElement: integer, input size: integer) -> integer
{ IS: Data 'L' telah disortir berdasarkan ID. 'targetElement' berisi ID yang ingin dicari. }
{ FS: Mengembalikan indeks dari user dengan ID yang cocok jika ditemukan, atau -1 jika tidak ditemukan. }

```

- Sequential search username

```

FUNCTION sequential_username (input L: ListUser, input/output targetName: char, input size: integer) -> integer
{ IS: Data 'L' berisi daftar user. 'targetName' adalah nama user yang ingin dicari. }
{ FS: Mengembalikan indeks dari user dengan username yang cocok jika ditemukan, atau -1 jika tidak ditemukan. }

```

- Search penyakit

```

PROCEDURE search_penyakit (input: L: ListUser, input/output targetPenyakit: char)
{ IS: Data 'L' sudah berisi daftar user lengkap, termasuk informasi penyakit untuk pasien. }
{ FS: Menampilkan daftar pasien dengan penyakit yang cocok dengan 'targetPenyakit'.
  Jika tidak ada pasien ditemukan, ditampilkan pesan bahwa tidak ditemukan. }

```

- Cari user

```

PROCEDURE cari_user (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login. }
{ FS: Jika pengguna adalah manajer dan data user ditemukan berdasarkan ID atau username,
  maka informasi user akan ditampilkan (selain manajer). Jika pengguna belum login atau bukan
  manajer,
  akan ditampilkan pesan yang sesuai dan pencarian tidak dilanjutkan. Jika ID/nama tidak ditemukan,
  akan ditampilkan pesan bahwa user tidak ditemukan. }

```

- Cari pasien

```

PROCEDURE cari_pasien (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login. }
{ FS: Jika pengguna adalah manajer dan data pasien ditemukan berdasarkan ID, nama, atau penyakit,
  maka informasi pasien akan ditampilkan. Jika pengguna belum login atau bukan manajer,
  akan ditampilkan pesan yang sesuai. Jika data tidak ditemukan atau user bukan pasien,
  akan ditampilkan pesan yang sesuai. }

```

- Cari dokter

```

PROCEDURE cari_dokter (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login. }
{ FS: Jika pengguna adalah manajer dan data dokter ditemukan berdasarkan ID atau nama,
  maka informasi dokter ditampilkan. Jika pengguna belum login atau bukan manajer,
  maka pencarian dibatalkan dengan menampilkan pesan yang sesuai. Jika user bukan dokter,
  maka ditampilkan pesan bahwa user bukan dokter. }

```

```

# DISCLAIMER: Tampilan/interface dibebaskan, berikut hanya contoh saja
# Kasus 1: Mencari data user (user bisa berarti dokter atau pasien) berdasarkan ID
>>> CARI_USER
Cari berdasarkan?

```

1. ID

2. Nama

>>> Pilihan: 1

>>> Masukkan nomor ID user: 3

Menampilkan pengguna dengan nomor ID 3...

ID	Nama	Role	Penyakit
3	Neroifa	Dokter	-

Kasus 2: User yang dicari tidak ditemukan

>>> **CARI_USER**

Cari berdasarkan?

1. ID

2. Nama

>>> Pilihan: 2

```

>>> Masukkan nama user: Kebin
Tidak ditemukan pengguna dengan nama Kebin!

# Kasus 3: Mencari data pasien
>>> CARI_PASIEN
Cari berdasarkan?
1. ID
2. Nama
3. Penyakit
>>> Pilihan: 3

>>> Masukkan nama penyakit: Maag

# Memungkinkan adanya pasien dengan penyakit Maag lebih dari satu
Urutkan berdasarkan?
1. ID
2. Nama
>>> Pilihan: 1

Urutan sort ID?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 2

Menampilkan pasien dengan penyakit Maag dengan ID terurut descendant...
ID | Nama      | Penyakit
-----
5  | Remon     | Maag
2  | Erik      | Maag

```

```

# Kasus 4: Mencari data dokter
>>> CARI_DOKTER
Cari berdasarkan?
1. ID
2. Nama
>>> Pilihan: 2

>>> Masukkan nama dokter: Alpin

Menampilkan dokter dengan nama Alpin...
ID | Nama
-----
6  | Alpin

```

9. F09 - Lihat Antrian

PROCEDURE lihat_semua_antrian(Input/output ruangan: ListRuang, input/output users: ListUser.)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login dan ingin melihat semua antrian pasien di tiap ruangan. }
{ FS: Jika pengguna belum login, maka sistem menampilkan pesan untuk login terlebih dahulu dan fungsi dibatalkan.
- Jika pengguna bukan manajer, maka sistem menampilkan pesan akses ditolak dan fungsi dibatalkan.
- Jika pengguna adalah manajer, maka sistem menampilkan daftar nomor ruangan yang tersedia.
- Untuk setiap ruangan yang memiliki dokter, sistem menampilkan detail ruangan termasuk

- kapasitas, nama dokter, pasien yang sedang berada dalam ruangan (sesuai kapasitas), dan pasien yang sedang mengantri setelah kapasitas terpenuhi.
- Jika ruangan kosong atau tidak ada pasien, sistem menampilkan pesan yang sesuai. }

10. F10 - Tambah Dokter

• Tambah Dokter

```
PROCEDURE tambah_dokter (input/output users: ListUser, input/output usernames: Set)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login sebagai manajer. Manajer ingin menambahkan dokter dengan menginput username dan password dari dokter.}
{ FS: Jika username dokter valid dan belum digunakan, maka dokter ditambahkan dan sistem menampilkan pesan berhasil.
- Jika pengguna masih belum login, maka sistem akan menampilkan pesan yang sesuai dan tidak akan bisa menambahkan dokter.
- Jika pengguna login bukan sebagai manajer, maka sistem akan menampilkan pesan akses ditolak dan tambah dokter akan dibatalkan.
- Jika username dokter mengandung karakter non-alfabet, maka tambah dokter dibatalkan dan sistem menampilkan pesan kesalahan.
- Jika username dokter sudah terdaftar, maka tambah dokter dibatalkan dan ditampilkan pesan bahwa nama sudah digunakan.
Jika input tidak valid atau kapasitas penuh, maka tambah dokter tidak dilanjutkan dan sistem menampilkan pesan yang sesuai}
```

```
# Kasus 1: Dokter bernama Budi belum ada
>>> TAMBAH_DOKTER
Username: Budi
Password: budi123
```

```
Dokter Budi berhasil ditambahkan!
>>> LOGOUT
```

```
Sampai jumpa!
```

```
>>> LOGIN
Username: Budi
Password: budi123
```

```
Selamat pagi Dokter Budi!
```

```
# Kasus 2: Dokter bernama Budi sudah ada
>>> TAMBAH_DOKTER
Username: Denis
Password: denis123
```

```
Sudah ada Dokter bernama Denis!
```

```
# Kasus 3: Pengguna Belum Login
>>> TAMBAH_DOKTER
```

```
Tidak ada pengguna yang sedang login. Silahkan login terlebih dahulu.
```

```
# Kasus 4: Pengguna Bukan Manajer
>>> TAMBAH_DOKTER
Akses ditolak. Anda bukan Manajer.
```

- Assign Dokter

PROCEDURE assign_dokter(input/output users: ListUser, input/output ruangan: ListRuangan)

{ IS: Program dalam keadaan berjalan. Pengguna sudah login sebagai manajer. Manajer ingin melakukan assign ruangan ke dokter tertentu yang belum memiliki ruangan.}

{ FS: Jika username dokter valid, maka dokter akan diassign ke salah satu ruangan dan sistem menampilkan pesan berhasil.

- Jika pengguna masih belum login atau login bukan sebagai manajer, maka sistem akan menampilkan pesan yang sesuai dan tidak akan bisa melakukan assign dokter.
- Jika username dokter tidak ditemukan karena belum ada sebelumnya, maka assign dokter dibatalkan dan sistem menampilkan pesan kesalahan.
- Jika dokter sudah menempati ruangan tertentu, maka assign dokter akan gagal dan sistem akan menampilkan pesan kesalahan.
- Jika ruangan tertentu sudah ditempati dokter lain, maka assign dokter akan gagal dan sistem akan menampilkan pesan kesalahan }

```
# Kasus 1: Dokter bernama Budi belum ada  
# Kasus 1: Ruangan Kosong dan dokter belum di assign di ruang manapun
```

```
>>> ASSIGN_DOKTER
```

```
Username: Budi
```

```
Ruangan: 1
```

```
Dokter Budi berhasil diassign ke ruangan 1!
```

```
# Kasus 2: Ruangan Kosong dan dokter sudah di assign di ruang lain
```

```
>>> ASSIGN_DOKTER
```

```
Username: Budi
```

```
Ruangan: 2
```

```
Dokter Budi sudah diassign ke ruangan 1!
```

```
# Kasus 3: Ruangan tidak kosong dan dokter belum di assign di ruang manapun
```

```
>>> ASSIGN_DOKTER
```

```
Username: Adi
```

```
Ruangan: 11
```

```
Dokter Budi sudah menempati ruangan 1!
```

```
Silakan cari ruangan lain untuk dokter Adi.
```

```
# Kasus 4: Ruangan tidak kosong dan dokter sudah di assign di ruang lain
```

```
>>> ASSIGN_DOKTER
```

```
Username: Ahuy
```

```
Ruangan: 1
```

```
Dokter Ahuy sudah menempati ruangan 3!
```

```
Ruangan 1 juga sudah ditempati dokter Budi!
```

11. F11 - Diagnosis

- Identifikasi Penyakit

FUNCTION identifikasi_penyakit (input/output pasien: Pasien) → **static character**

{ IS: Data pasien (suhu, tekanan darah, dll.) sudah terisi.}

{ FS: Mengembalikan nama penyakit jika gejala pasien cocok dengan kriteria penyakit tertentu, atau NULL jika tidak ditemukan.}

```
>>> IDENTIFIKASI_PENYAKIT
```

```
# Penyakit tidak ditemukan
> NULL
# Penyakit ditemukan
> namaPenyakit
```

- Diagnosis Pasien

```
PROCEDURE diagnosis_pasien (input/output userDokter: User)
{ IS: Program dalam keadaan berjalan, user dokter sedang login, dan memiliki pasien dalam antrian ruangan.}
{ FS: Bergantung pada kondisi pasien, sistem akan:
  - Mengubah status pasien (butuhDiberiObat atau butuhPulang).
  - Menampilkan pesan sesuai hasil diagnosis.
  - Memastikan data pasien terupdate di sistem.
}
```

```
# Kasus 1: User belum login atau bukan dokter
>>> DIAGNOSIS
# Kasus belum login
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
# Kasus bukan dokter
ERROR: Hanya dokter yang bisa diagnosis!
```

```
# Kasus 2: Dokter tidak memiliki ruang atau tidak ada pasien dalam antrian
>>> DIAGNOSIS
# Kasus dokter tidak memiliki ruang
[dr. Budi] Kamu belum memiliki ruangan.
# Kasus tidak ada pasien dalam antrian
[dr. Budi] Kamu lagi nggak ada pasien. Asik, free time!
```

```
# Kasus 3: Status pasien bukan butuhDiagnosa
>>> DIAGNOSIS
# Kasus butuhDiberiObat
[dr. Budi] Pasien Denis udah didiagnosa, tinggal dikasih obat aja.
# Kasus bukan butuhDiagnosa dan bukan butuhDiberiObat
[dr. Budi] Pasien Denis tidak perlu didiagnosis lagi.
```

```
# Kasus 4: Pasien terdiagnosa tidak memiliki penyakit
>>> DIAGNOSIS
+-----+
|           DIAGNOSA           |
+-----+
[dr. Budi] Pasien @Denis sehat banget! Dijamin kuat salto keliling kota!
```

```
# Kasus 5: Pasien terdiagnosa penyakit
# Contoh kasus penyakit: Influenza
>>> DIAGNOSIS
+-----+
|           DIAGNOSA           |
+-----+
[dr. Budi] Pasien @Denis terdiagnosa mengidap penyakit: Influenza
[dr. Budi] Jangan lupa untuk diobatin ya!
HELP: Untuk mengobati pasien @Denis, ketik NGOBATIN!
```

12. F12 - Ngobatin

- Cari Obat

```
FUNCTION cari_obat (input/output namaPenyakit: constant character) → PenyakitObatEntry
{ IS: Nama penyakit valid dan database penyakitObatMap tersedia.}
{ FS:
    - Jika penyakit ditemukan: Mengembalikan pointer ke PenyakitObatEntry yang berisi daftar obat.
    - Jika tidak ditemukan: Mengembalikan NULL.
}

# Kasus 1: Penyakit ditemukan
>>> CARI_OBAT
# Penyakit tidak ditemukan
> NULL
# Penyakit ditemukan
# Contoh: Entry untuk Hipertensi
> &penyakitObatMap[2];
```

- Ngobatin

```
PROCEDURE ngobatin (input/output currUser: User, input/output users: User, input banyakUser: integer, input/output ruangan: ListRuangan)
{ IS: User dokter sedang login dan user adalah dokter. Data ruangan dan pasien dalam antrian tersedia.}
{ FS:
    - Jika valid, status pasien diubah menjadi butuhMinumObat dan daftar obat disimpan.
    - Pesan sesuai kondisi ditampilkan.
}
```

```
# Kasus 1: User belum login atau bukan dokter
>>> NGOBATIN
# Kasus belum login
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
# Kasus bukan dokter
ERROR: Hanya dokter yang punya kemampuan mengobati pasien!
```

```
>>> NGOBATIN
# Kasus dokter tidak memiliki ruangan
[dr. Budi] Kamu belum memiliki ruangan.
# Kasus tidak ada pasien dalam antrian
[dr. Budi] Kamu lagi nggak ada pasien. Ngobatin siapa?
```

```
# Kasus 3: Status pasien tidak valid
>>> NGOBATIN
# Kasus butuhDiagnosa
[dr. Budi] Pasien Denis belum didiagnosis. Diagnosis dulu sebelum mengobati!
# Kasus butuhPulang
[dr. Budi] Pasien Denis sudah sembuh, tidak perlu diobati lagi.
# Kasus butuhMinumObat
[dr. Budi] Pasien Denis sudah diberi obat, tidak perlu diobati lagi.
```

```
# Kasus 4: Tidak ada obat untuk penyakit
>>> NGOBATIN
[dr. Budi] Tidak ada obat untuk penyakit Hipertensi!
```

```
# Kasus 5: Berhasil memberi resep obat
>>> NGOBATIN
[dr. Budi] Resep untuk @Denis (Penyakit: Demam Berdarah):
    1. Paracetamol
```

- | |
|---|
| <ol style="list-style-type: none"> 2. Oralit 3. Vitamin C |
|---|

13. F13 - Aku boleh pulang ga, dok?

- Mengubah data pasien menjadi 0 (reset seperti semula)

<pre>PROCEDURE make_default_pasien (input/output: User: Pasien) { IS: Pasien dinyatakan sembuh dan dipersilakan pulang oleh dokter.} { FS: Mengembalikan semua data pasien menjadi semula atau menjadi 0 kembali seperti awal registrasi.}</pre>

- Mengecek kondisi pasien/akah pasien boleh pulang atau tidak

<pre>PROCEDURE pulangdok (input/output: currUser: User, input/output: users: User, input/output: Obat: daftarObat)</pre>

<pre>{ IS: User yang login pada program adalah pasien. Keadaan pasien dalam sudah atau belum didiagnosis ditentukan oleh PROCEDURE Diagnosis. Apabila sudah didiagnosis, terdapat urutan obat yang diresepkan dokter, serta obat-obat yang sudah diminum oleh pasien disusun dalam bentuk Stack. Pasien juga harus mengantri kembali pada satu ruangan}</pre>
--

<pre>{ FS: Program tidak memperbolehkan pasien pulang apabila:</pre>
--

- | |
|--|
| <ul style="list-style-type: none"> - Belum menerima diagnosis dari dokter - Pasien tidak berada di urutan pertama untuk masuk ke ruangan. - Ada obat yang belum dihabiskan - Urutan obat yang diminum tidak sesuai resep dokter. |
|--|

<pre>Apabila semua kondisi telah dipenuhi, program menampilkan pesan yang memperbolehkan pasien untuk pulang.}</pre>
--

<pre># KASUS 1: Belum ada user yang login >>> PULANGDOK</pre>
--

<pre>ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.</pre>

<pre># KASUS 2: User yang login bukan pasien >>> PULANGDOK</pre>

<pre>ERROR: Hanya pasien yang bisa konsultasi pada dokter lagi!</pre>

<pre># KASUS 3: Ruangan pasien tidak ada >>> PULANGDOK</pre>

<pre>ERROR: Ruangan pasien tidak ditemukan!</pre>

<pre># KASUS 4: Pasien bukanlah antrian terdepan di ruangan dokter. >>> PULANGDOK</pre>
--

<pre>>> Maaf, kamu bukan pasien paling depan di antrian. Tunggu giliranmu ya.</pre>

<pre># KASUS 5: Pasien belum didiagnosa oleh dokter >>> PULANGDOK</pre>
--

<pre>[@nama user pasien] Loh... kamu belum didiagnosa dokter sama sekali, jangan buru-buru pulang!</pre>
--

<pre>HELP: Untuk daftar gunakan DAFTAR_CHECKUP dan untuk cek antrianmu gunakan LIHAT_ANTRIAN!</pre>

```

# KASUS 6: Pasien belum meminum atau menghabiskan obat yang diberikan
>>> PULANGDOK

Kamu belum menghabiskan seluruh obat yang diberikan. Yuk minum obatmu dulu!

# KASUS 7: Pasien sudah menghabiskan obat, tapi minum urutan obat yang salah
>>> PULANGDOK

Dokter sedang memeriksa keadaanmu...

Maaf, kamu kayaknya masih belum bisa pulang, masih sakit ya?

Urutan peminuman obat yang diharapkan:
[Obat] -> [Obat] -> [Obat] -> [Obat]

Urutan obat yang kamu minum
[Obat] -> [Obat] -> [Obat] -> [Obat]

Kunjungi dokter untuk meminta penawar yang sesuai ya!

```

```

# KASUS 8: Pasien sudah menghabiskan obat dan semua syarat sudah dilakukan. Pasien
bisa pulang.
>>> PULANGDOK

Dokter sedang memeriksa keadaanmu...

Yay Selamat! Kamu sudah dinyatakan sembuh oleh dokter. Silahkan pulang dan semoga
sehat selalu

# Kembali ke menu utama dan pasien meninggalkan ruangan. Data pasien direset.

```

14. F14 - Daftar Check Up

- Validasi Float

```

FUNCTION validasi_float (input nilai: float, input/output namaVar: constant character) → boolean
{ IS: Variabel nilai dan namaVar telah diberikan.
{ FS:
    - Jika nilai positif: Mengembalikan true dan tidak menampilkan pesan.
    - Jika nilai negatif: Menampilkan pesan error dan mengembalikan false.
}

```

```

>>> VALIDASI_FLOAT
# Input valid
> true
# Input tidak valid
> Suhu harus positif!
> false

```

- Validasi Integer

```

FUNCTION validasi_integer (input nilai: integer, input/output namaVar: constant character) →
boolean
{ IS: Variabel nilai dan namaVar telah diberikan.
{ FS:
    - Jika nilai positif: Mengembalikan true dan tidak menampilkan pesan.
    - Jika nilai negatif: Menampilkan pesan error dan mengembalikan false.
}

```

```

}

>>> VALIDASI_INTEGER
# Input valid
> true
# Input tidak valid
> Suhu harus positif!
> false

```

- Display Dokter

```

PROCEDURE display_dokter (input/output users: User, input banyakUser: integer, input/output
ruangan: ListRuangan)
{ IS: Array users dan struktur ruangan tersedia dan valid. Variabel banyakUser berisi jumlah user yang
valid.}
{ FS:
    - Menampilkan daftar dokter beserta informasi ruangan dan jumlah antrian.
    - Jika ditemukan data dokter tidak valid, menampilkan pesan error yang spesifik.
}

```

```

# Kasus 1: Data dokter valid
>>> DISPLAY_DOKTER
1. dr. Budi (Ruangan 3) - Total Antrian: 5
2. dr. Ani (Belum memiliki ruangan)
3. dr. Citra (Ruangan 1) - Total Antrian: 2

```

```

# Kasus 2: Data dokter tidak valid
>>> DISPLAY_DOKTER
ERROR: Data dokter tidak valid untuk dr. Johan!
2. dr. Ani (Ruangan 2) - Total Antrian: 3

```

```

# Kasus 3: Tidak ada dokter terdaftar
>>> DISPLAY_DOKTER
Berikut adalah daftar dokter yang tersedia:
[Tidak ada dokter yang tersedia]

```

- Daftar Check Up

```

PROCEDURE daftar_check_up (input/output currUser: User, input/output users: User, input
banyakUser: integer, input/output ruangan: ListRuangan)
{ IS: Program berjalan, user pasien sedang login. Data pasien valid.}
{ FS: Bergantung pada kondisi, sistem akan:
    - Memasukkan pasien ke antrian dokter yang dipilih.
    - Mengupdate posisiAntrian pasien.
    - Menampilkan pesan sukses/gagal beserta detail antrian.
}

```

```

# Kasus 1: User belum login atau bukan pasien
>>> DAFTAR_CHECKUP
# Kasus belum login
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
# Kasus bukan dokter
ERROR: Hanya pasien yang bisa daftar check-up.

```

```

# Kasus 2: Pasien sudah terdaftar di antrian
>>> DAFTAR_CHECKUP

```

```

# Sudah dalam antrian
[@Andi] Kamu sudah terdaftar di antrian!
Silakan selesaikan check-up yang sudah terdaftar terlebih dahulu.
# Sudah di ruangan lain
[@Andi] Kamu sudah berada di ruangan 3 (atau antrian ruangan 3).

# Kasus 3: Input data medis tidak valid
>>> DAFTAR_CHECKUP
# Proses berulang hingga input valid
Suhu tubuh (Celcius): -5
Suhu harus positif!
Suhu tubuh (Celcius): 37.5 // Lanjut setelah input valid

# Kasus 4: Dokter tidak valid/tidak ditemukan
>>> DAFTAR_CHECKUP
ERROR: Dokter tidak ditemukan!

# Kasus 5: Berhasil daftar ke ruangan/antrian
>>> DAFTAR_CHECKUP
# Dokter memiliki ruangan
>> Pasien berhasil didaftarkan ke ruangan 2.
>> Pendaftaran berhasil! <<
[@Andi] Kamu terdaftar pada antrian dr. Budi di ruangan 2.
[@Andi] Posisi antrian Kamu: 3
# Dokter belum punya ruangan
ERROR: Dokter Budi belum memiliki ruangan. Mendaftarkan ke antrian...
>> Pendaftaran berhasil! <<
[@Andi] Kamu terdaftar pada antrian dr. Budi.

```

15. F15 - Antrian Saya

```

PROCEDURE cek_antrian_saya (input/output user: User, input/output users: User, input banyakUser: integer, input/output ruangan: ListRuangan)
{ IS: User pasien sedang login. Data pasien valid. Struktur ruangan dan users tersedia.}
{ FS:
    - Menampilkan status antrian pasien atau pesan error sesuai kondisi
    - Tidak mengubah data sistem
}

# Kasus 1: User belum login atau bukan pasien
>>> ANTRIAN_SAYA
# Kasus belum login
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
# Kasus bukan dokter
ERROR: Hanya pasien yang bisa melihat antrian!

# Kasus 2: Pasien tidak terdaftar di antrian
>>> ANTRIAN_SAYA
[@Andi] Kamu nih belum terdaftar dalam antrian check-up (berkata dengan nada lemah lembut gemulai). Daftar dulu coba :D

HELP: Silakan daftar terlebih dahulu dengan command DAFTAR_CHECKUP.

# Kasus 3: Pasien sudah di ruangan dokter
>>> ANTRIAN_SAYA
[@Budi] Kamu lagi di ruangan dokter, loh. Dokternya ga lagi tidur kan?

# Kasus 4: Pasien dalam antrian

```

```

>>> ANTRIAN_SAYA
+-----+
|           LIHAT ANTRIAN           |
+-----+
[@Budi] Status antrian Anda:
Dokter: dr. Citra
Ruangan: 3
Posisi antrian: 2 dari 5

# Kasus 5: Error sistem
>>> ANTRIAN_SAYA
# Ruangan tidak ditemukan
ERROR: Ruangan tidak ditemukan!
# Antrian kosong
ERROR: Antrian ruangan 2 kosong!
# Dokter tidak ditemukan
ERROR: Dokter pasien Andi tidak ditemukan!

```

16. F16 - Minum Obat

```

PROCEDURE minum_obat (input/output: currUser: User, input/output: Obat: daftarObat, output:
Pasien: perutPasien)
{ IS: Terdapat daftar obat yang diresepkan oleh pasien hasil proses ngobatin dokter. Status pasien =
ButuhMinumObat}
{ FS: Jika nomor obat yang dipilih valid:
  - Obat tersebut dihapus dari daftar obat
  - Obat dimasukkan ke dalam Stack perutPasien
  - Program menampilkan bahwa obat berhasil diminum
  - Status pasien diubah ke butuhPulang
Jika nomor obat yang dipilih tidak valid (tidak ada di daftar obat):
  - Program menampilkan bahwa obat tidak tersedia
  - Tidak terjadi perubahan pada daftarObat dan perutPasien
Jika pasien meminum obat dengan urutan yang salah:
  - Obat yang diminum dihapus dari daftarObat
  - Obat dimasukkan ke dalam Stack perutPasien
  - Status pasien diubah menjadi butuhPenawar
  - Pasien tidak akan bisa meminum obat yang lain karena status berubah ke butuhPenawar}

```

```

# Tampilan daftar obat
>>> MINUM_OBAT

===== DAFTAR OBAT =====
Urutan obat sesuai resep dokter:
[Obat] -> [Obat] -> [Obat] -> [Obat]

Obat yang harus diminum
1. [Obat]
2. [Obat]
3. [dst...]

# Tampilan daftar obat yang sudah diminum

Obat yang sudah diminum:
[Obat] -> [Obat] -> [Obat] -> [Obat]

# Tampilan daftar obat yang sudah diminum (tapi perut kosong)
Obat yang sudah diminum:

```

```
Belum ada obat yang diminum

# KASUS 1: Belum ada user yang login
>>> MINUM_OBAT

ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.

# KASUS 2: User yang login bukan pasien
>>> MINUM_OBAT

Eits...pasien doang yang bisa minum obat!

# KASUS 3: Pasien mau minum obat atau sebenarnya butuh penawar
>>> MINUM_OBAT

Yuk minum obatmu dulu..
Kamu punya [jumlah] obat yang harus diminum.

//else
Kamu lagi ga perlu minum obat kok.

# KASUS 4: Pasien salah minum obat
>>> MINUM_OBAT

Kamu merasa makin gak enak badan...kayaknya kamu salah minum urutan obat deh..

Dok! Butuh obat penawar!!

# Kasus 5: Apabila obat yang dipilih tidak ada
Pilih obat untuk diminum: [pilihan Obat]

Obatnya gak diresepin buat kamu!

# Kasus 6: Apabila obat yang dipilih ada
Pilih obat untuk diminum: [pilihan Obat]

GLEKGLEKGLEK... [Obat] berhasil diminum!

# KASUS 7: Pasien harus minum obat penawar dahulu karena salah minum urutan obat
>>> MINUM_OBAT

Kamu lagi keracunan gara-gara urutan minumnya salah.. Mau minum penawarnya dulu gak?
(y/n):
Kamu punya [jumlah] obat yang harus diminum.

//bila pilih ya (y)
Oke minum obat penawarnya dulu ya!.

//bila pilih tidak (n) -> kembali ke Kasus 3

//else
Input tidak valid. Silakan coba lagi.

# KASUS 8: Pasien selesai minum semua obat tapi masih butuh penawar
>>> MINUM_OBAT

Kamu sudah minum semua obat, tapi masih butuh penawar.

# KASUS 8: Pasien selesai minum semua obat secara benar dan bisa pulang
```

```
>>> MINUM_OBAT
```

```
Kamu sudah minum semua obat yang diberikan dokter. Silahkan pulang ya!
```

17. F17 - Minum Penawar

PROCEDURE minumPenawar (input/output: currUser: User, input/output: Obat: daftarObat, output: Pasien: perutPasien)

{ IS: Pasien memiliki daftar obat. Obat-obat yang sudah diminum oleh pasien dan masuk ke dalam perut disusun dalam bentuk Stack. Pasien minum urutan obat yang salah, sehingga mengakibatkan pasien kembali sakit dan harus minum obat penawar. }

{ FS: Jika Stack perut tidak kosong maka:

- Obat yang terakhir diminum pasien (top of stack) di perutPasien dikeluarkan
- Obat tersebut ditambahkan kembali ke daftar obat di akhir
- Program menampilkan bahwa obat telah dipindahkan ke inventory kembali

Jika Stack konsumsi kosong:

- Tidak ada perubahan pada perutPasien dan inventory
- Program menampilkan bahwa belum ada obat yang dimakan}

```
# KASUS 1: Belum ada user yang login
```

```
>>> MINUM_PENAWAR
```

```
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
```

```
# KASUS 2: User yang login bukan pasien
```

```
>>> MINUM_PENAWAR
```

```
Eits...pasien doang yang bisa minum penawar!
```

```
# Kasus 3: Belum ada obat yang diminum
```

```
>>> MINUM_PENAWAR
```

```
Perut kosong!! Belum ada obat yang dimakan.
```

```
# Kasus 4: Penawar diminum dan obat mulai dikeluarkan
```

```
>>> MINUM_PENAWAR
```

```
Uwekkk!!! [Obat] keluar dan kembali ke inventory
```

```
# Kasus 5: Urutan obat masih belum sesuai
```

```
>>> MINUM_PENAWAR
```

```
Urutan obat masih belum sesuai resep dokter! Muntahkan lagi satu obat...
```

```
# Kasus 6: Urutan obat sudah kembali semula dan pasien bisa minum obat lagi
```

```
>>> MINUM_PENAWAR
```

```
Urutan obat terdepan sudah sesuai resep dokter!
```

```
# Pasien bisa lanjut minum obat menggunakan MINUM_OBAT
```

18. F18 - Exit

PROCEDURE exit_system()

{ IS: Program sedang berjalan. Pengguna diminta untuk memilih apakah ingin menyimpan perubahan sebelum keluar (y/n). }

{ FS: Jika input 'y' atau 'Y', sistem akan menjalankan prosedur penyimpanan dan keluar dari program.

Jika input 'n' atau 'N', program langsung keluar tanpa menyimpan.

Jika input tidak valid, pengguna akan diminta untuk mengulang hingga memasukkan input yang valid ('y' atau 'n'). }

```
# Kasus 1: User ingin keluar dari program
```

```
>>> EXIT
```

```
Apakah Anda mau keluar dari rumah sakit? (y/n) y
```

```
# Keluar program
```

```
# Kasus 2: User tidak keluar dari program
```

```
>>> EXIT
```

```
Apakah Anda mau keluar dari rumah sakit? (y/n) n
```

```
# Menjalankan fungsi-fungsi lain dan tidak keluar dari program
```

19. B05 - Dead or Alive

FEATURE Dead or Alive

{ IS:

- Pasien terdaftar dalam sistem rumah sakit
- Pasien memiliki status butuhMinumObat atau butuhPenawar
- Stack perutPasien berisi obat yang telah diminum

}

{ FS:

- Mengurangi nyawa pasien jika salah minum obat
- Me-reset nyawa jika menggunakan penawar dengan benar
- Mengeluarkan pasien dari sistem jika nyawa habis

}

```
# Pasien baru terdaftar: F02
```

{ FS:

- pasien→sisaNyawa = 3
- pasien→status = butuhDiagnosa

}

```
# Pasien salah minum obat: F16
```

{ FS:

- pasien→sisaNyawa berkurang 1
- pasien→status = butuhPenawar
- Tampilkan peringatan

}

```
# Pasien menggunakan penawar: F17
```

{ FS:

- pasien→sisaNyawa = 3
- pasien→status = butuhMinumObat
- Obat dikembalikan ke inventory

}

```
# Nyawa pasien habis
```

{ FS:

- pasien→status = meninggalDunia

- ```

 - Dikeluarkan dari antrian
 - currUser = NULL
}

```

## 20. B06 - Mainin Antrian

### **FEATURE Mainin Antrian**

Akses : Pasien

Deskripsi: Memungkinkan pasien memanipulasi posisi antrian dokter

- Skip Antrian

**PROCEDURE skip\_antrian** (input/output user: User, input/output: ruangan, input banyakUser: integer)

- ```

{ IS:
    - User pasien sedang login
    - Pasien terdaftar dalam antrian dokter
    - Ruangan dokter valid
}
{ FS:
    - Posisi pasien dipindah ke depan antrian (setelah pasien di dalam ruangan)
    - Posisi antrian lain menyesuaikan
    - Data antrian di-update
}

```

```

# Kasus 1: Skip antrian berhasil
# Pasien pindah ke posisi MAX_PASIEN_RUANGAN + 1
# Antrian sebelumnya bergeser mundur
>>> SKIP_ANTRIAN
[@Budi] Kamu berhasil pindah ke posisi pertama antrian!

```

```

# Kasus 2: Pasien sudah berada di posisi terdepan
>>> SKIP_ANTRIAN
Kamu sudah berada di posisi pertama antrian.

```

```

# Kasus 3: Pasien sudah berada di dalam ruangan
>>> SKIP_ANTRIAN
Kamu sedang berada di dalam ruangan, tidak bisa skip antrian.

```

```

# Kasus 4: Error
>>> SKIP_ANTRIAN
# User belum login
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
# User bukan pasien
ERROR: Hanya pasien yang bisa melewati antrian!
# User belum mendaftar check-up
ERROR: Kamu belum terdaftar dalam antrian check-up.

```

F. DESAIN KAMUS DATA

1. F01 - Login

KAMUS LOKAL

```
Login, found_user, valid_password : Boolean  
username, password, role : String  
i : Integer
```

2. F02 - Register Pasien

KAMUS LOKAL

```
is_valid_input, is_alpha_only, is_username_unique: Boolean  
username, password : String
```

3. F03 – Logout

KAMUS LOKAL

```
is_logged_in : Boolean
```

4. F04 – Lupa Password

KAMUS LOKAL

```
username, kode_unik_input, kode_unik_asli, password_baru : String  
i : integer  
Is_valid_input, is_kode_sesuai : Boolean
```

5. F05 – Menu & Help

KAMUS LOKAL

```
username : String
```

6. F06 – Denah Rumah Sakit

KAMUS LOKAL

```
r : pointer to Ruangan  
i : integer  
global ruangan : ListRuang  
global currUser: pointer to User
```

7. F07 – Lihat User

KAMUS LOKAL

```
i: integer  
ca: char  
cb: char  
cmp: integer  
tmp: array of integer
```

```
view: integer
L: ListUser
pilihan1: integer
pilihan2: integer
idx: array of integer
len: integer
```

8. F08 – Cari User

```
KAMUS LOKAL
L: ListUser
left, right, middle, pilihan, idTarget: integer
usernameTarget: string
penyakitTarget: string
found: integer
userIdx, len: integer
u: User
role_s: string
penyakit: string
```

9. F09 – Lihat Antrian

```
KAMUS LOKAL
ruangan: Pointer to ListRuangan
num: integer
users: Pointer to ListUser
r: Ruangan
i: integer
global currUser: pointer to User
countPasien: integer
counter: integer
current: address
```

10. F10 - Tambah Dokter

```
KAMUS LOKAL
username, password : String
nomor_ruangan, ruangan_idx : integer
dokter_baru : User
dokter_found, ruangan_found : boolean
global current_user : pointer to User
global users : ListUser
global ruangan : ListRuangan
```

11. F11 - Diagnosis

```
KAMUS LOKAL
```

```
namaPenyakit : static character
i : integer
p : Penyakit
dokter: Dokter
r: Ruangan
current : address
userPasien : User
pasien : Pasien
penyakit : character
```

12. F12 - Ngobatin

KAMUS LOKAL

```
dokter : Dokter
ruanganDokter : Ruangan
i : integer
current : address
pasien : Pasien
userPasien : User
```

13. F13 - Aku boleh pulang ga, dok 😊?

KAMUS LOKAL

```
userPasien : User
global ruangan : ListRuangan
pasien : Pasien
ruangan Dokter : Ruangan
daftarObat : Obat
i,jumlahObat : integer
perutPasien : Stack
```

14. F14 - Daftar Check-Up

KAMUS LOKAL

```
i, j, banyakDokter, pilihan, idxDokter : integer
dokter, dokterPilihan : Dokter
ruanganDokter, r : Ruangan
pasien, result : Pasien
curr : address
userDokter : User
```

15. F15 - Antrian Saya!

KAMUS LOKAL

```
pasien : Pasien
found, posisi, idxRuang, i, totalAntrian : integer
curr : address
```

dokterPasien : Dokter

16. F16 - Minum Obat

KAMUS LOKAL

currUser, userPasien : User
perutPasien : Stack
pasien : Pasien
pilihanObat, valid, idxObat, i : int
konfirmasi : character

17. F17 - Minum Penawar

KAMUS LOKAL

currUser, userPasien : User
perutPasien : Stack
pasien : Pasien
perutPasien : Stack
urutanBenar, i, n : int
obatTerakhir : Obat

18. F18 - Exit

KAMUS LOKAL

username: String
input: Character

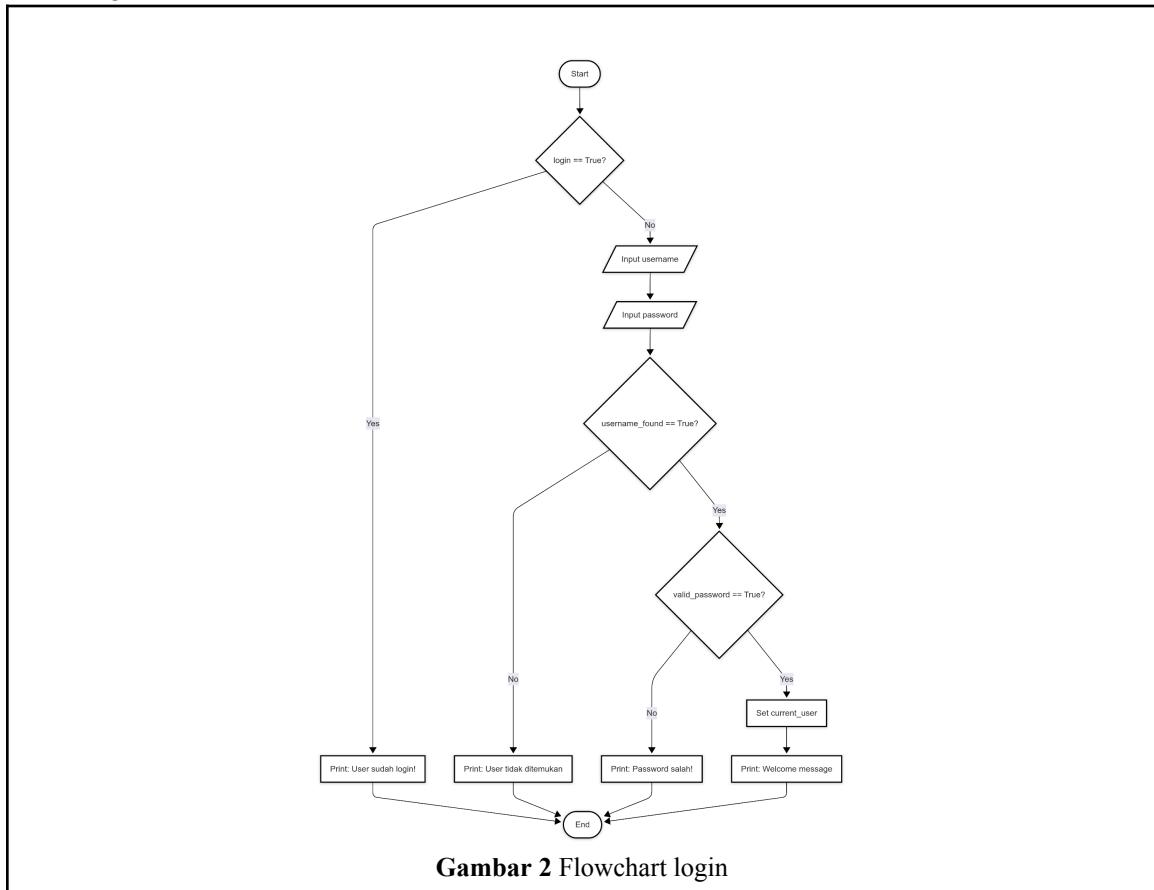
19. B06 - Mainin Antrian

KAMUS LOKAL

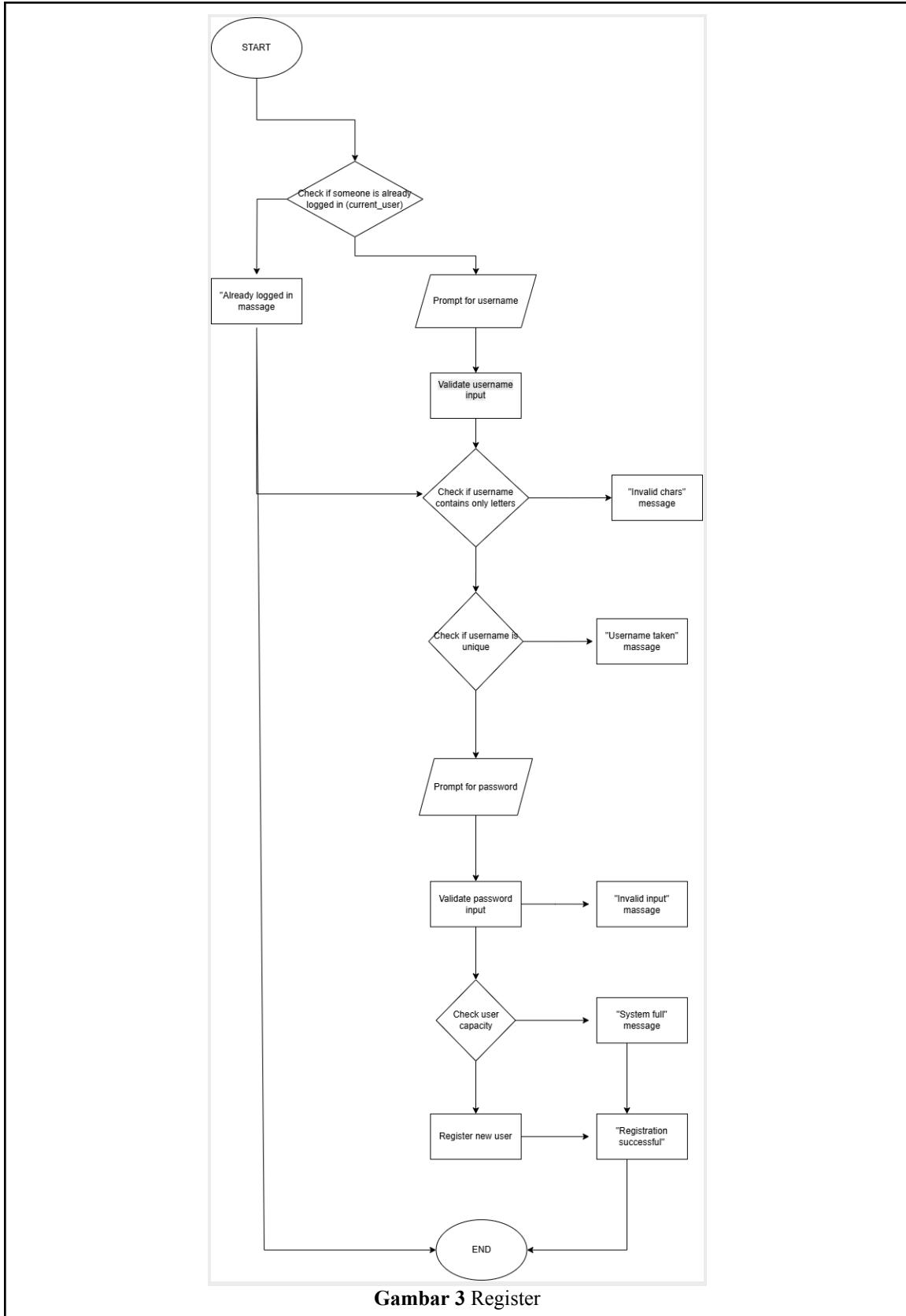
pasien : Pasien
idxRuang, maxDalamRuang, pos : integer
antrian : AntrianDokter
prev, curr, prevskip, firstAntrian : address

G. DESAIN DEKOMPOSISI ALGORITMIK DAN FUNGSIONAL PROGRAM

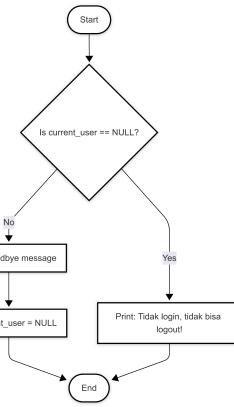
1. F01 - Login



2. F02 - Register Pasien

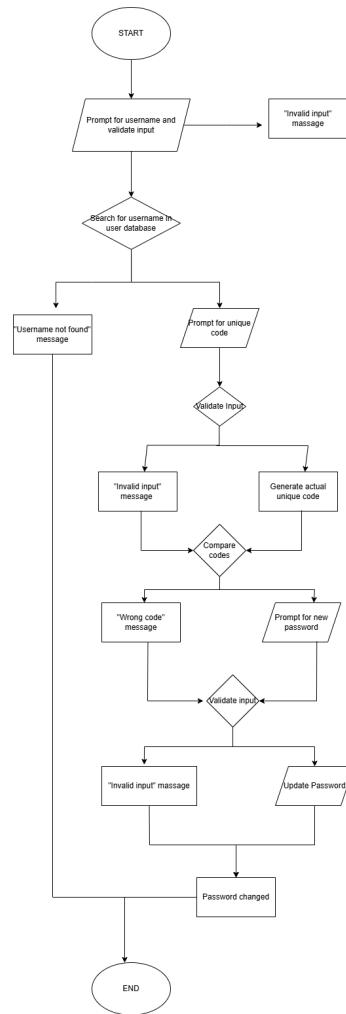


3. F03 - Logout



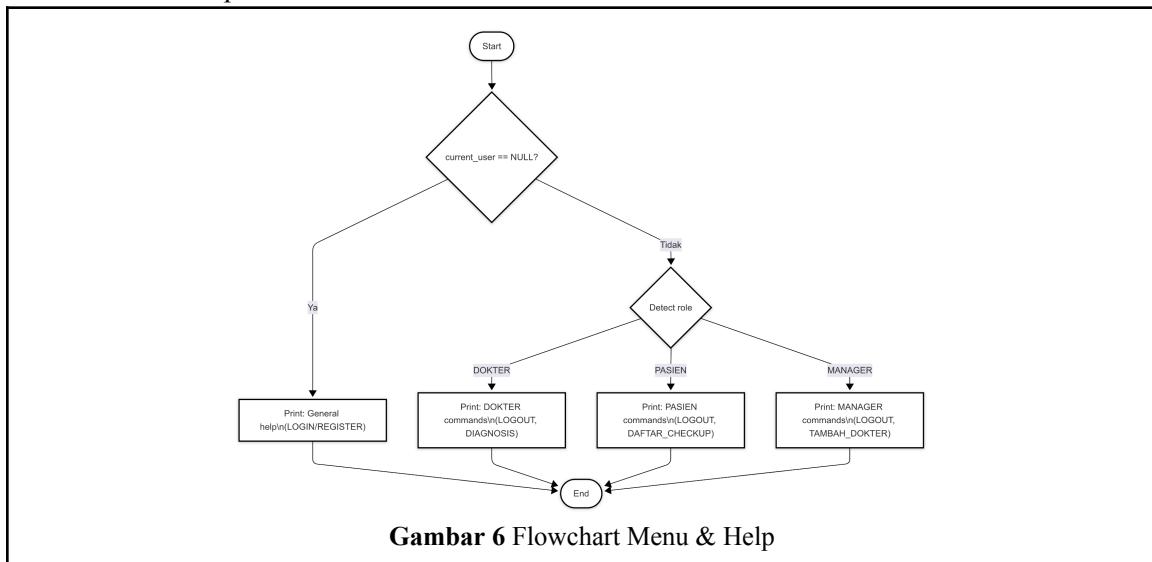
Gambar 4 Flowchart Logout

4. F04 - Lupa Password

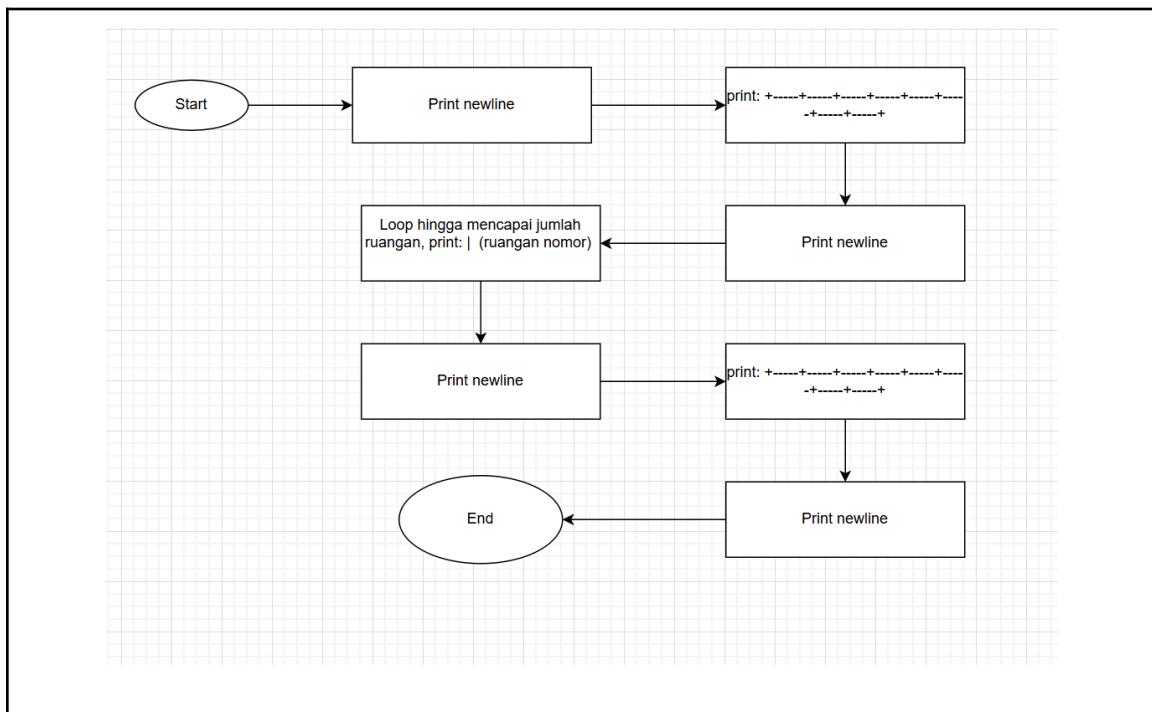


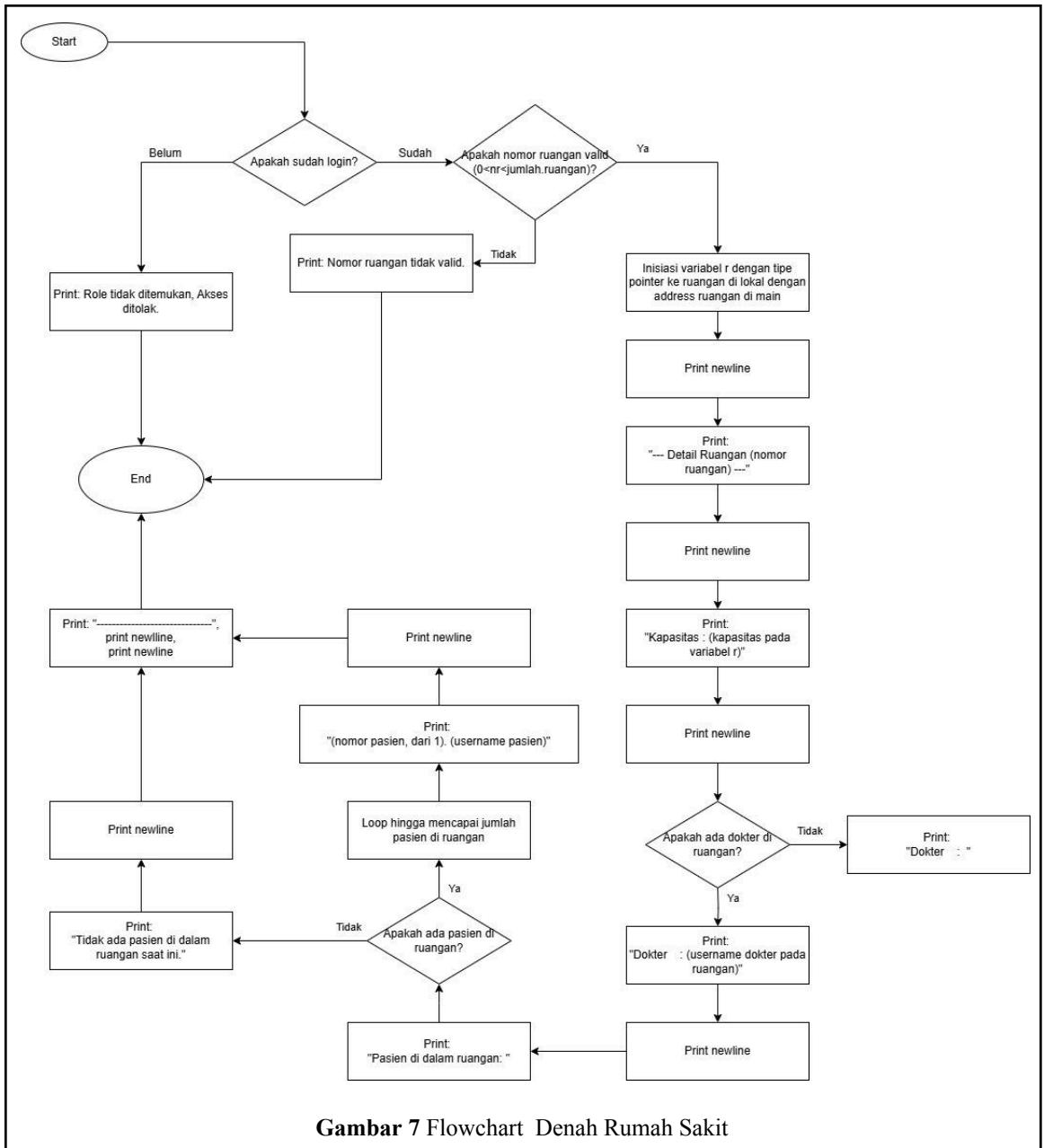
Gambar 5 Flowchart Lupa Password

5. F05 - Menu & Help

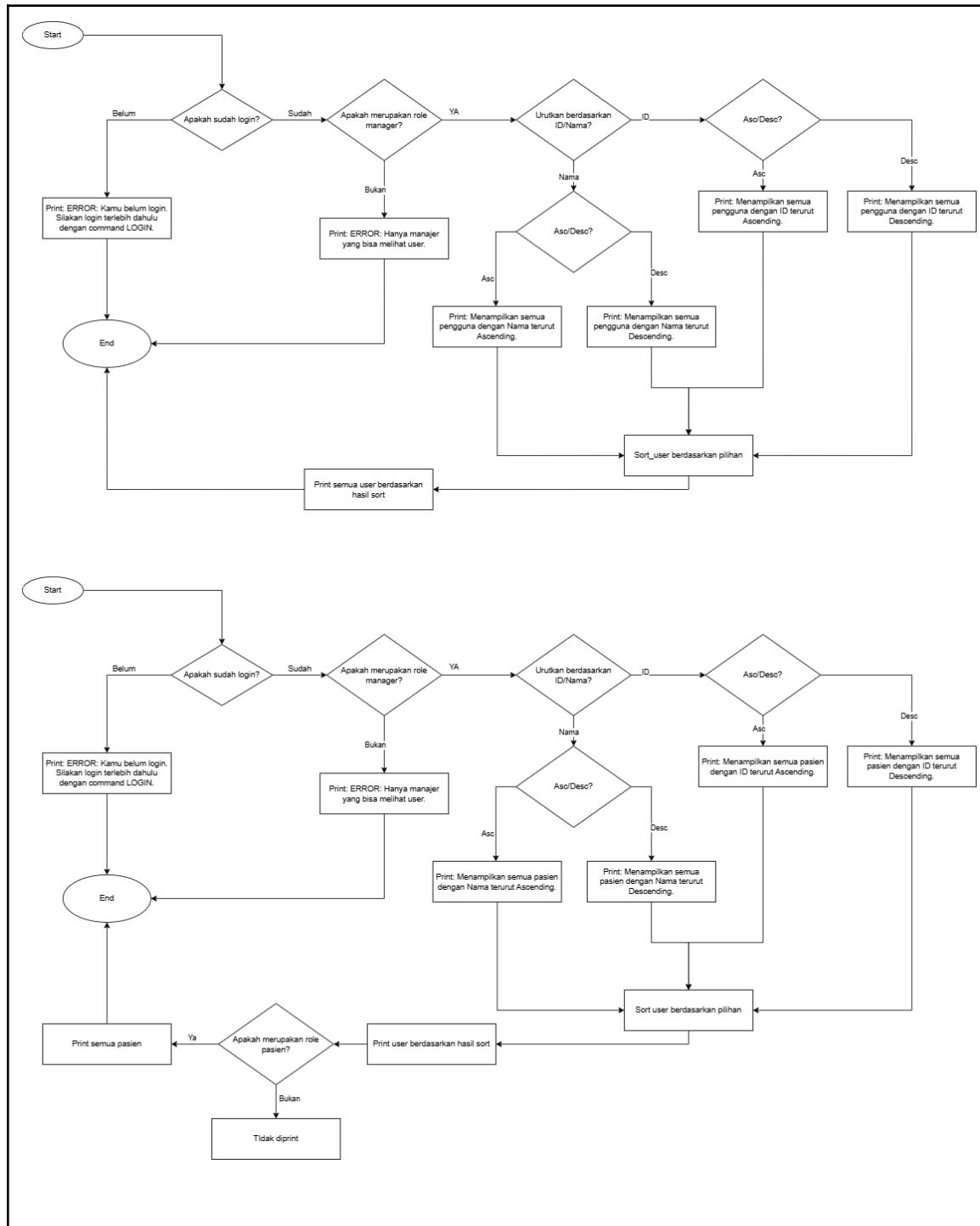


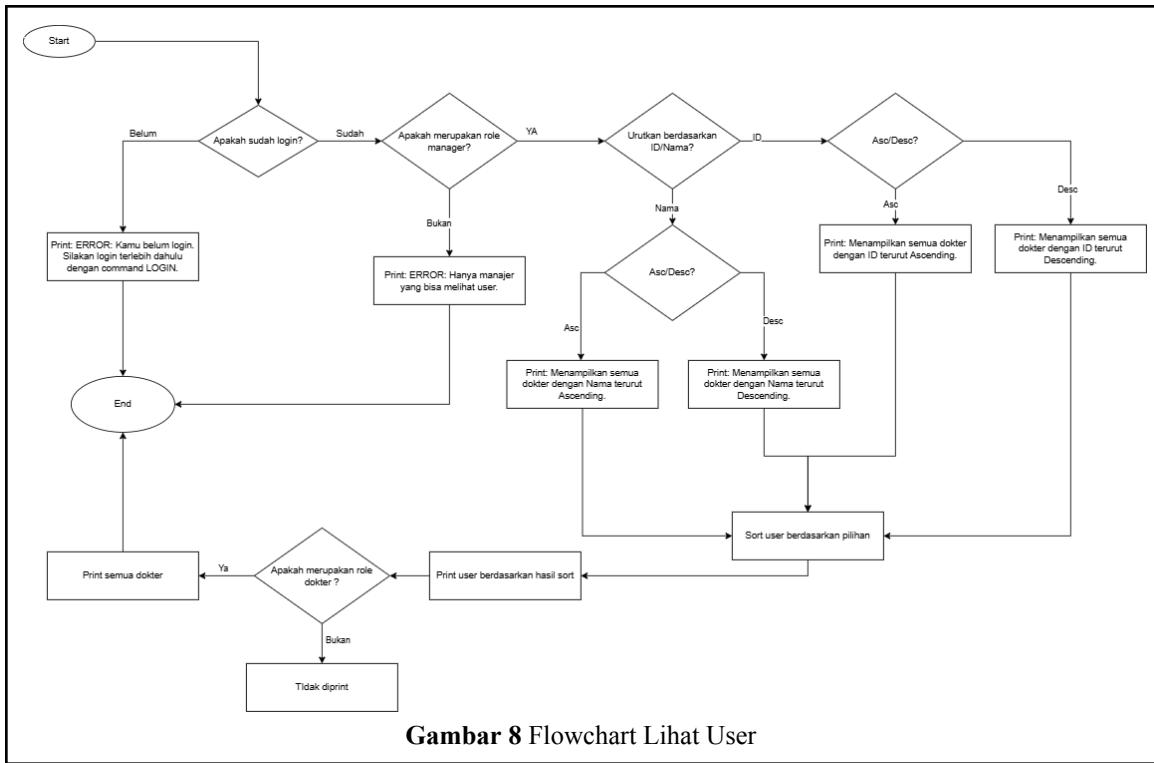
6. F06 - Denah Rumah Sakit



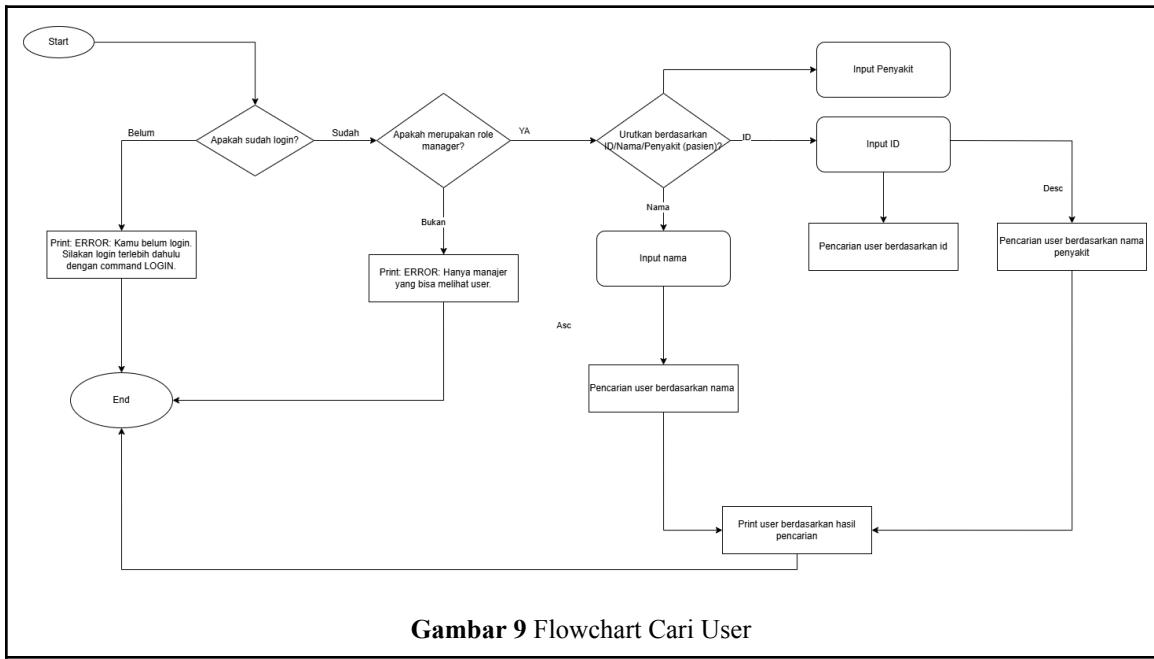


7. F07 - Lihat User

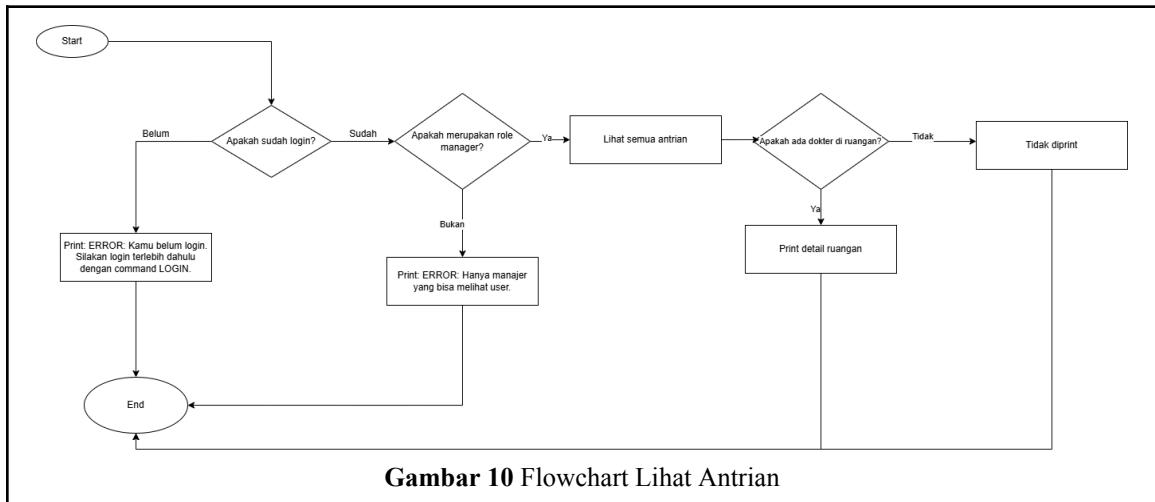




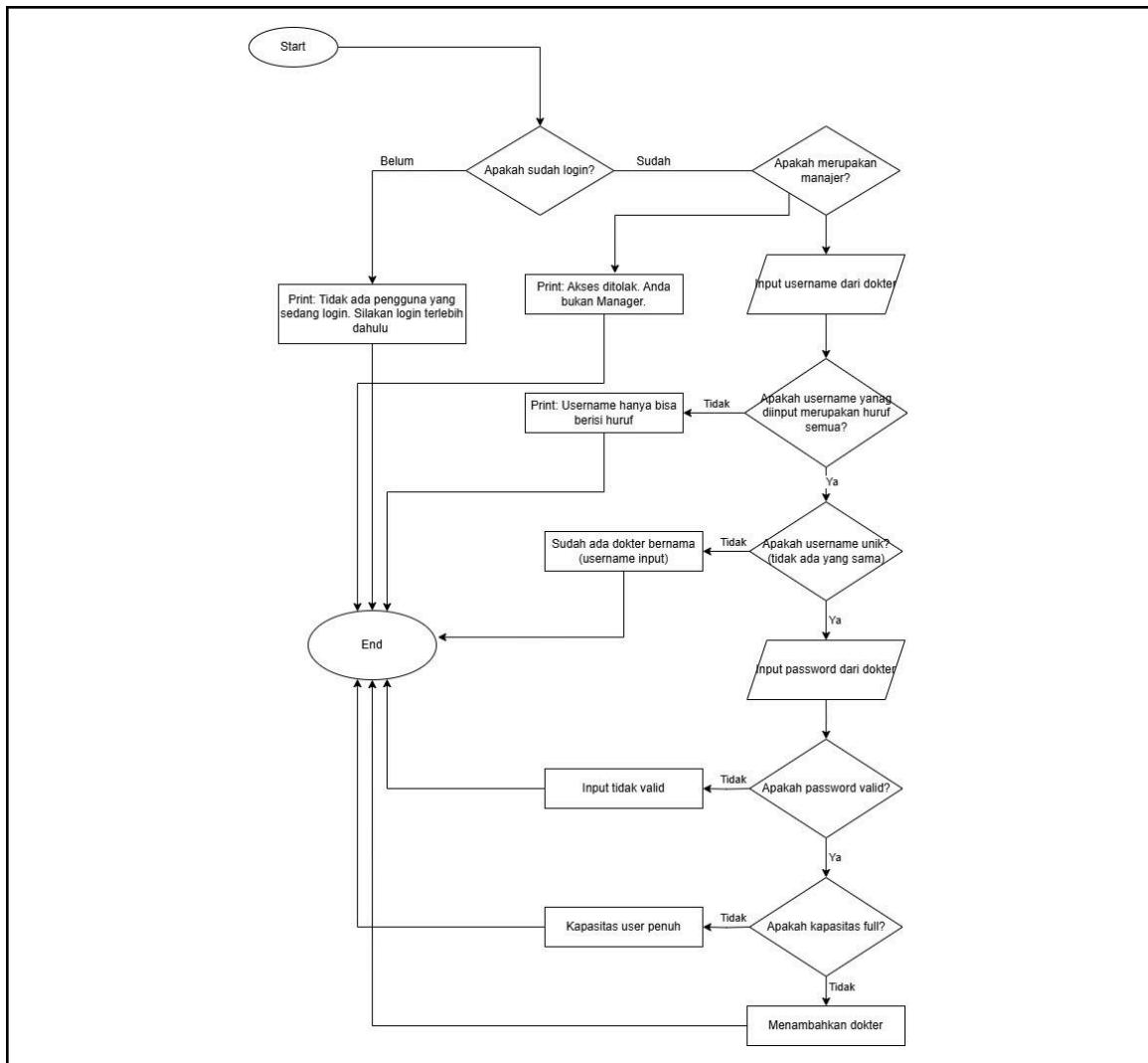
8. F08 - Cari User

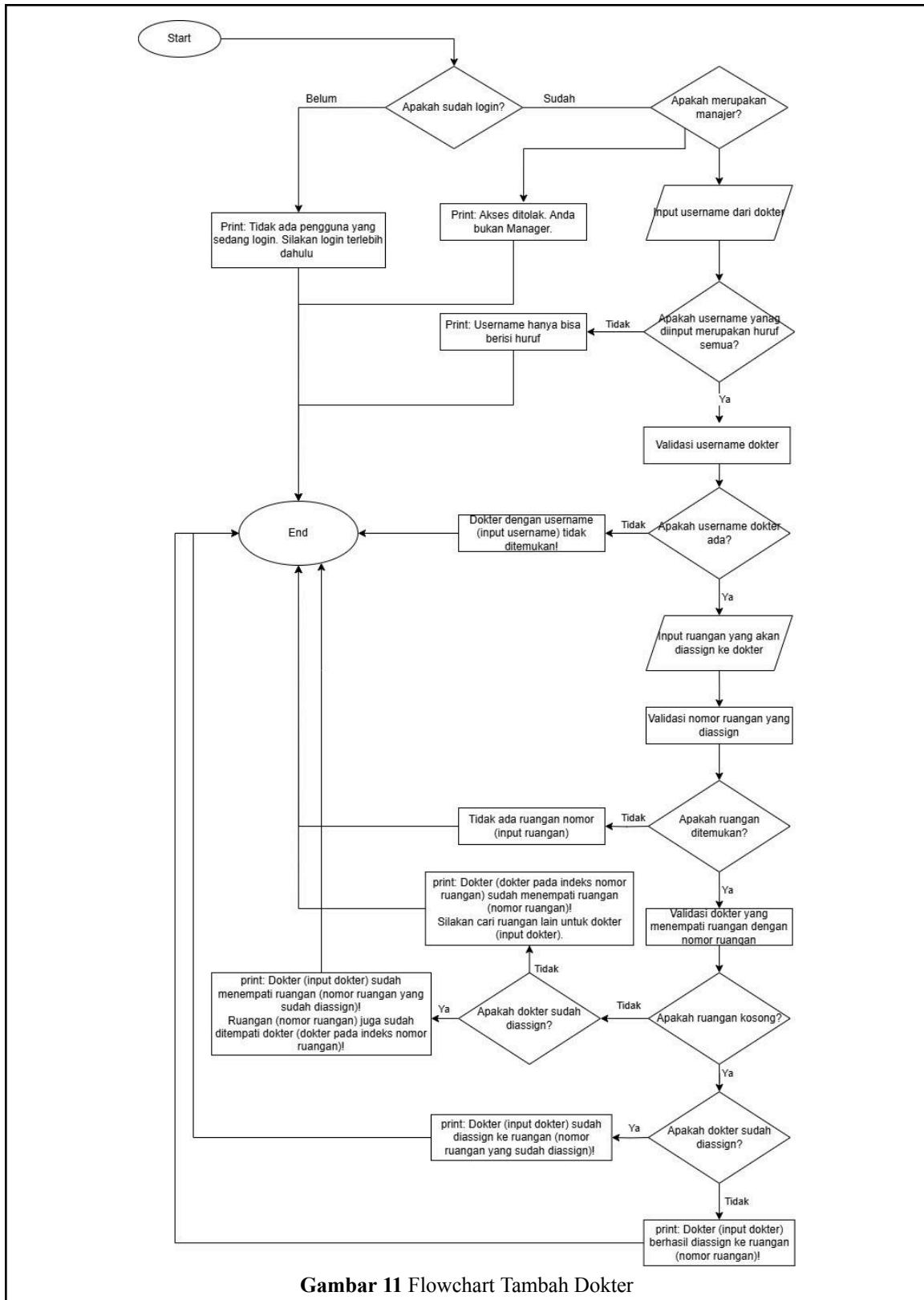


9. F09 - Lihat Antrian



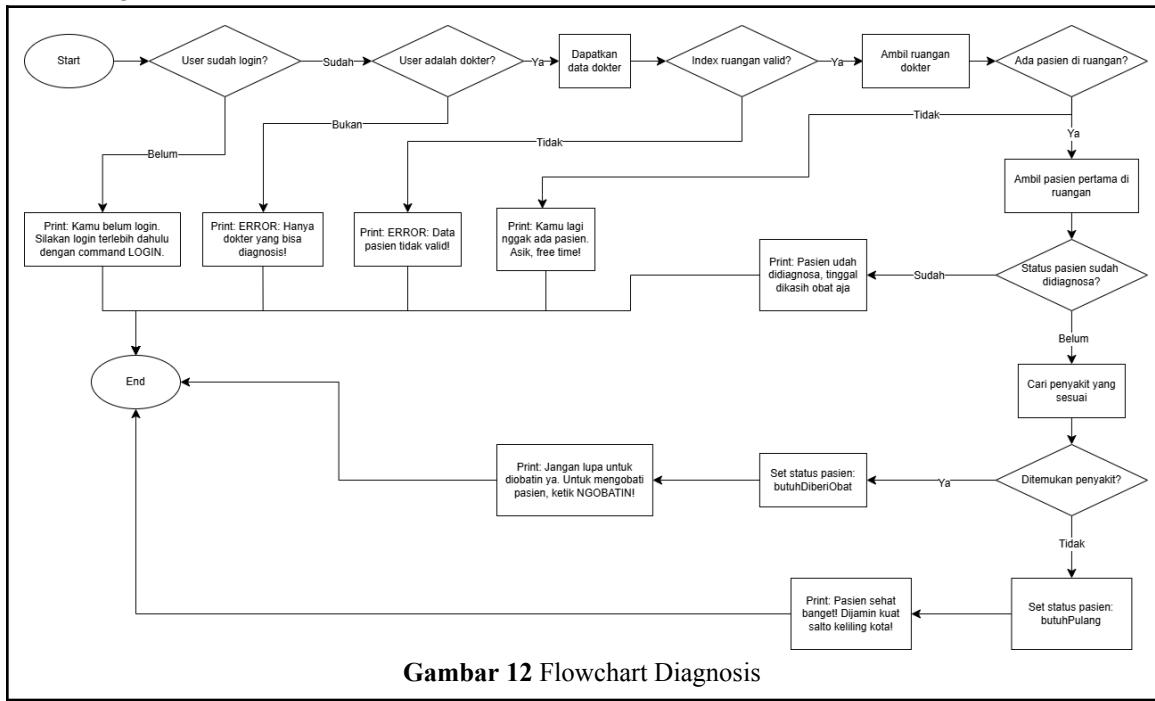
10. F10 - Tambah Dokter



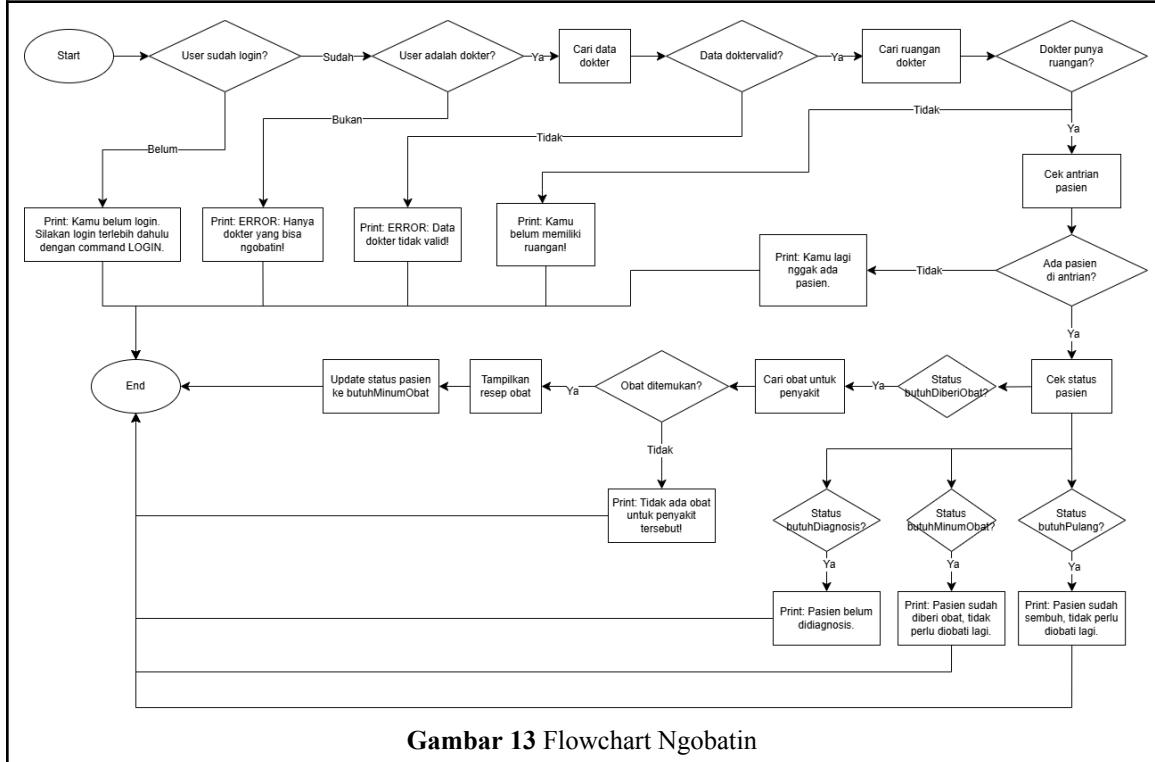


Gambar 11 Flowchart Tambah Dokter

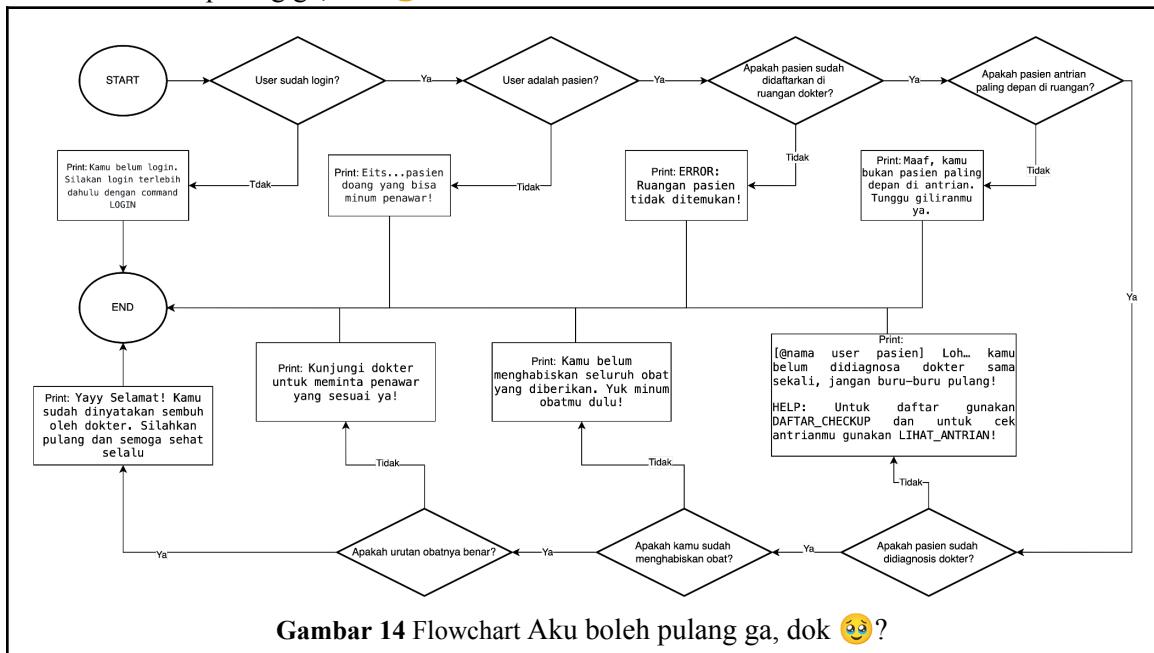
11. F11 - Diagnosis



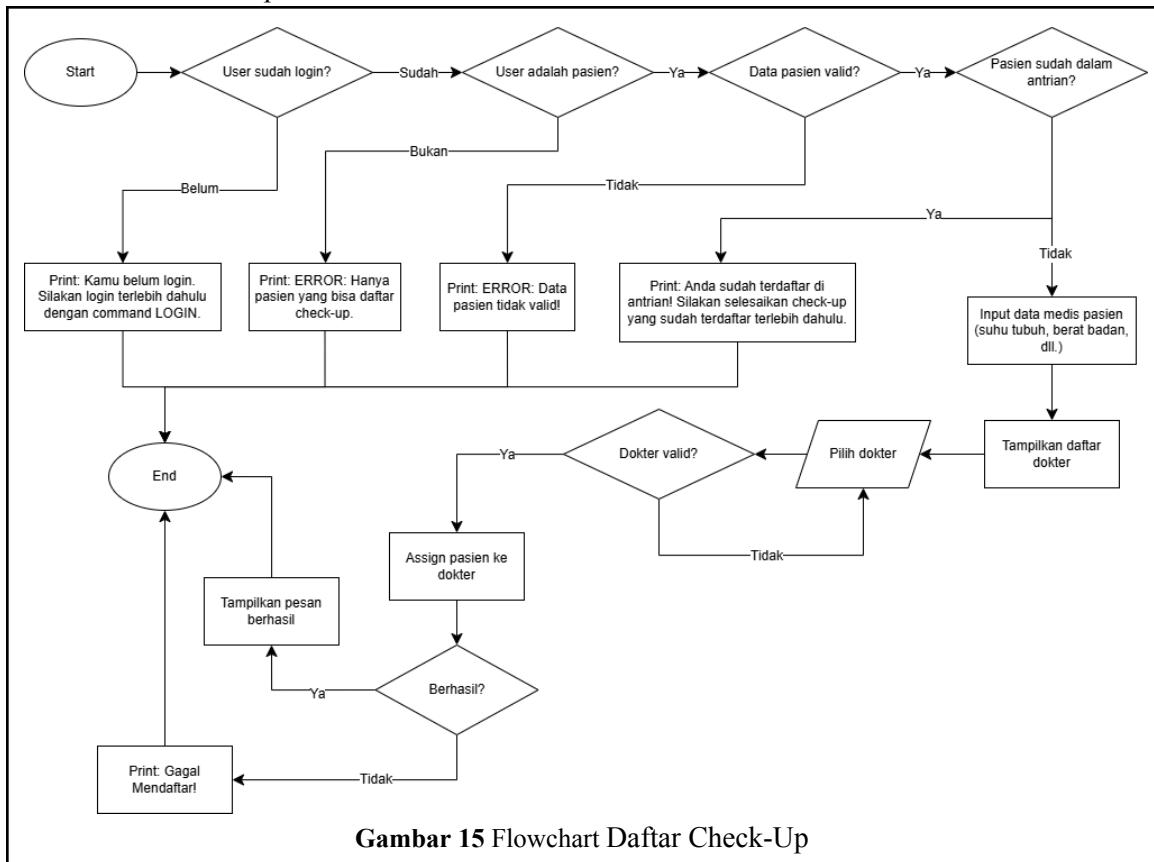
12. F12 - Ngobatin



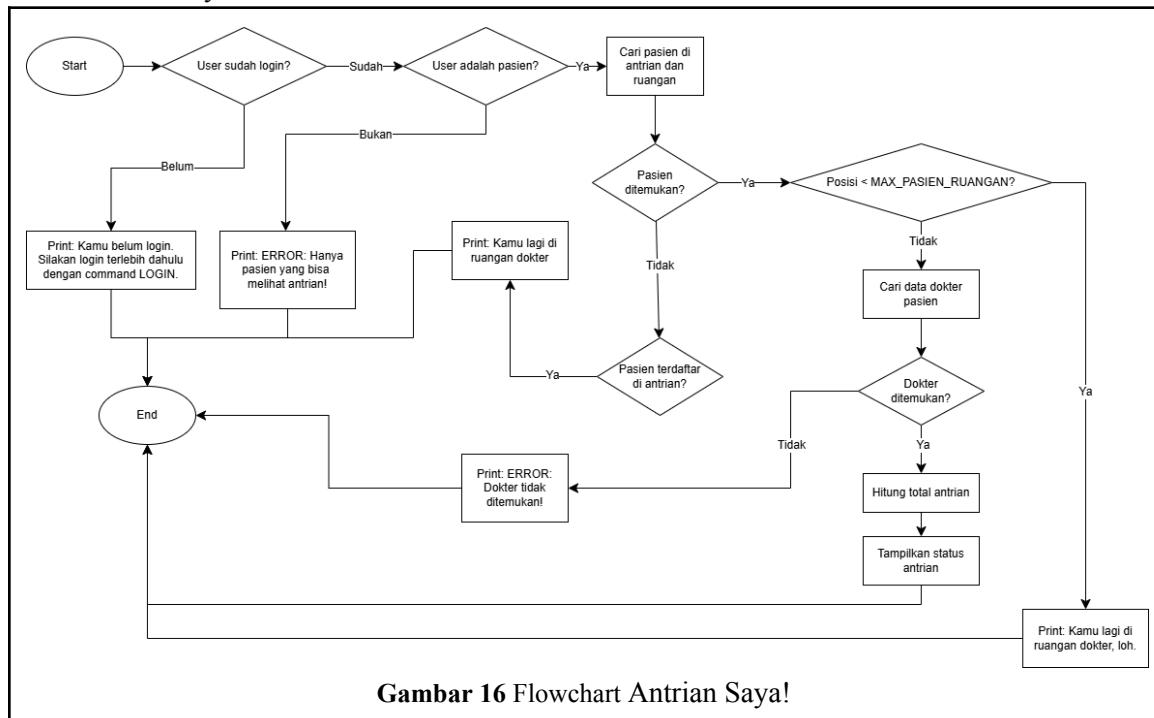
13. F13 - Aku boleh pulang ga, dok 😊?



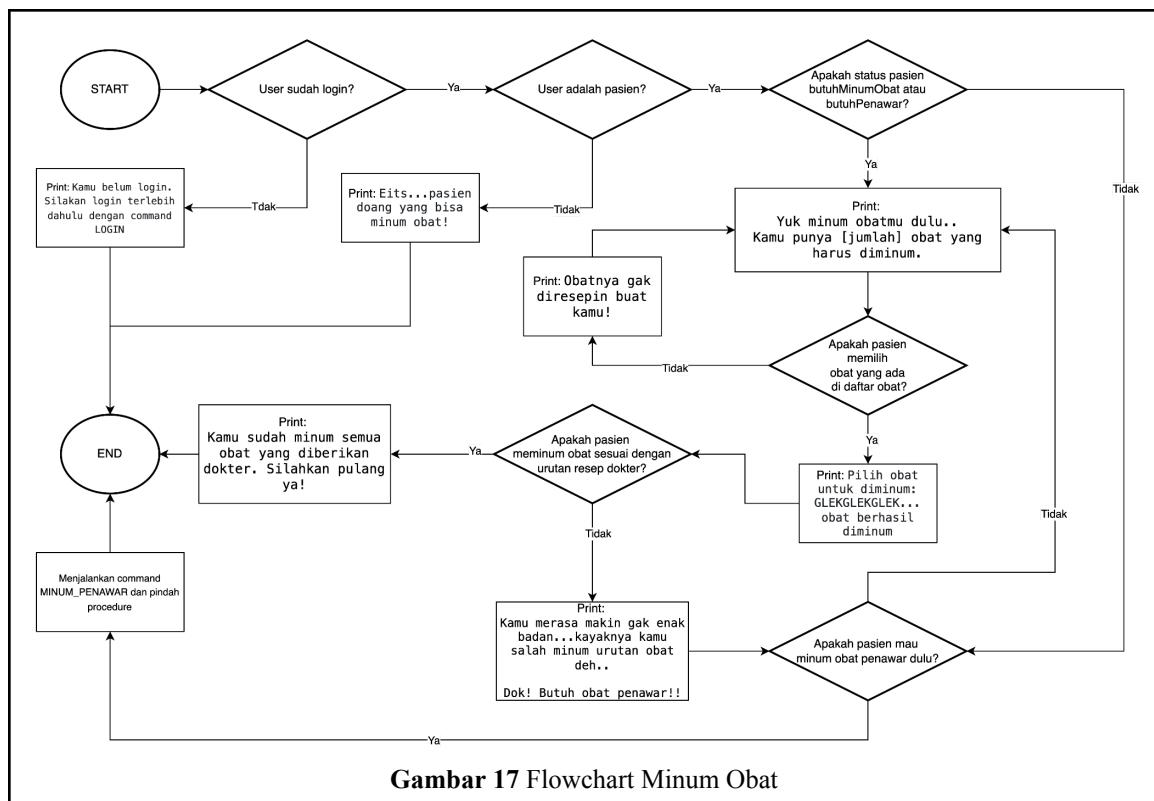
14. F14 - Daftar Check-Up



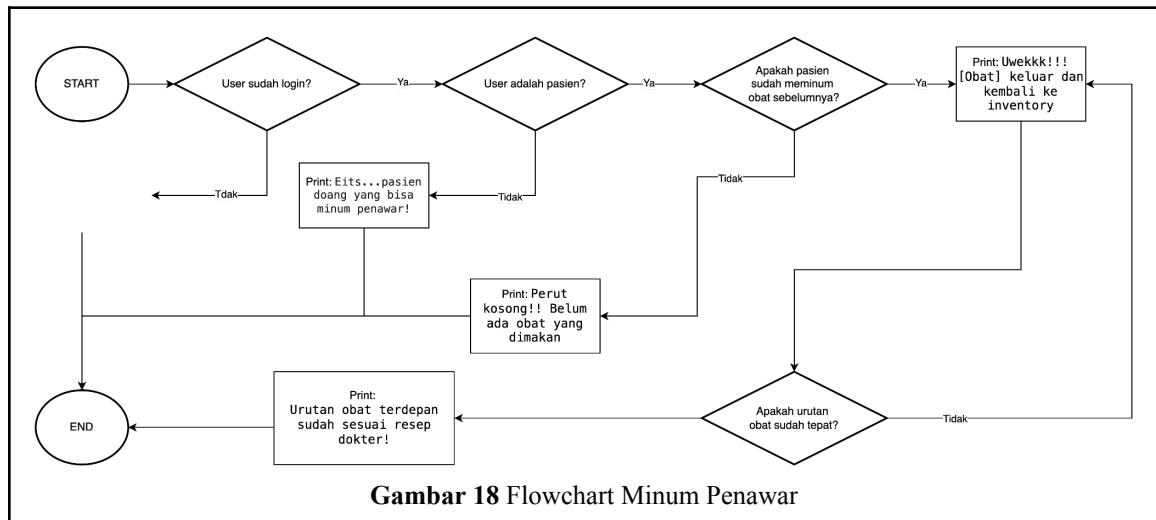
15. F15 - Antrian Saya!



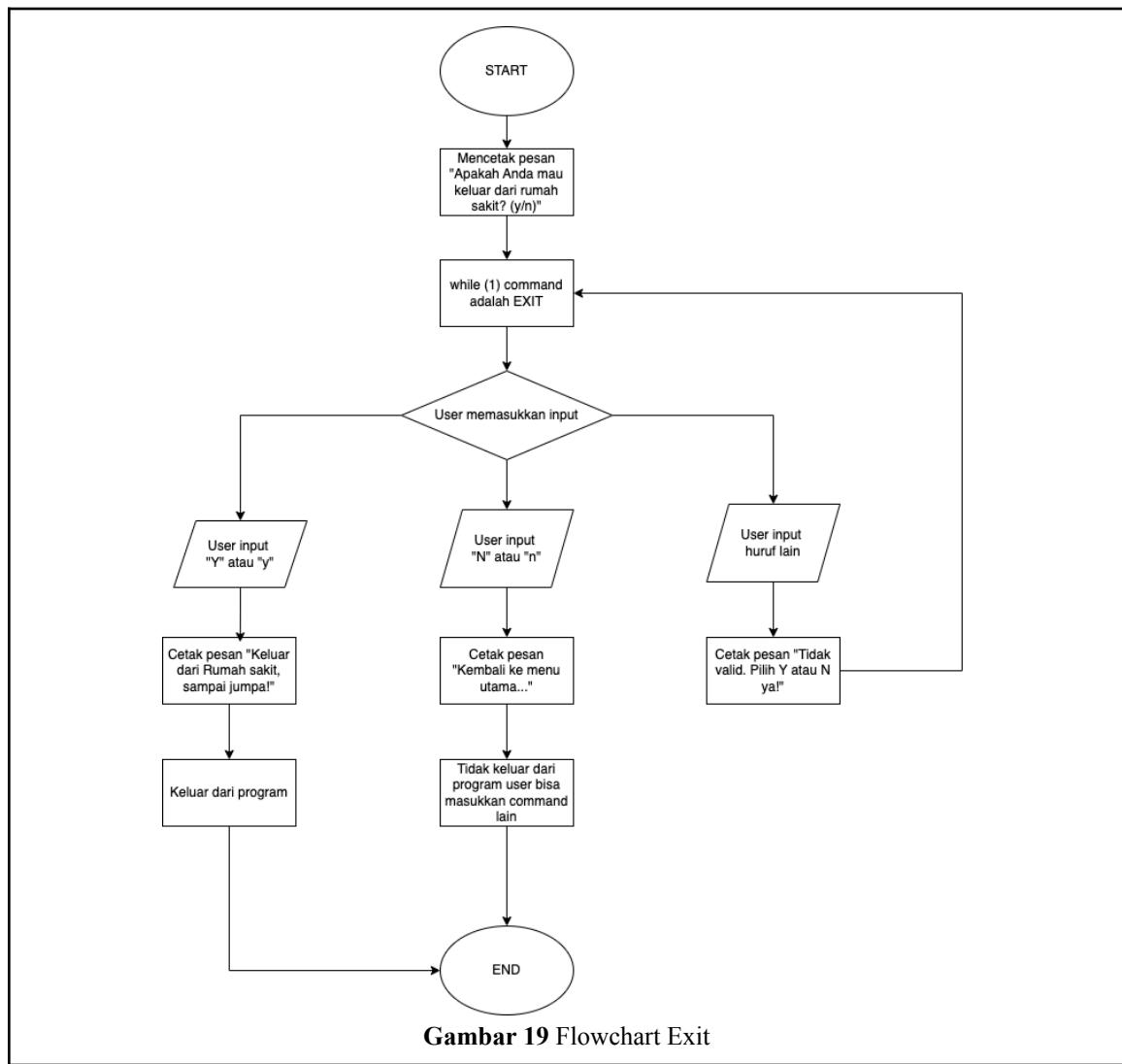
16. F16 - Minum Obat



17. F17 - Minum Penawar

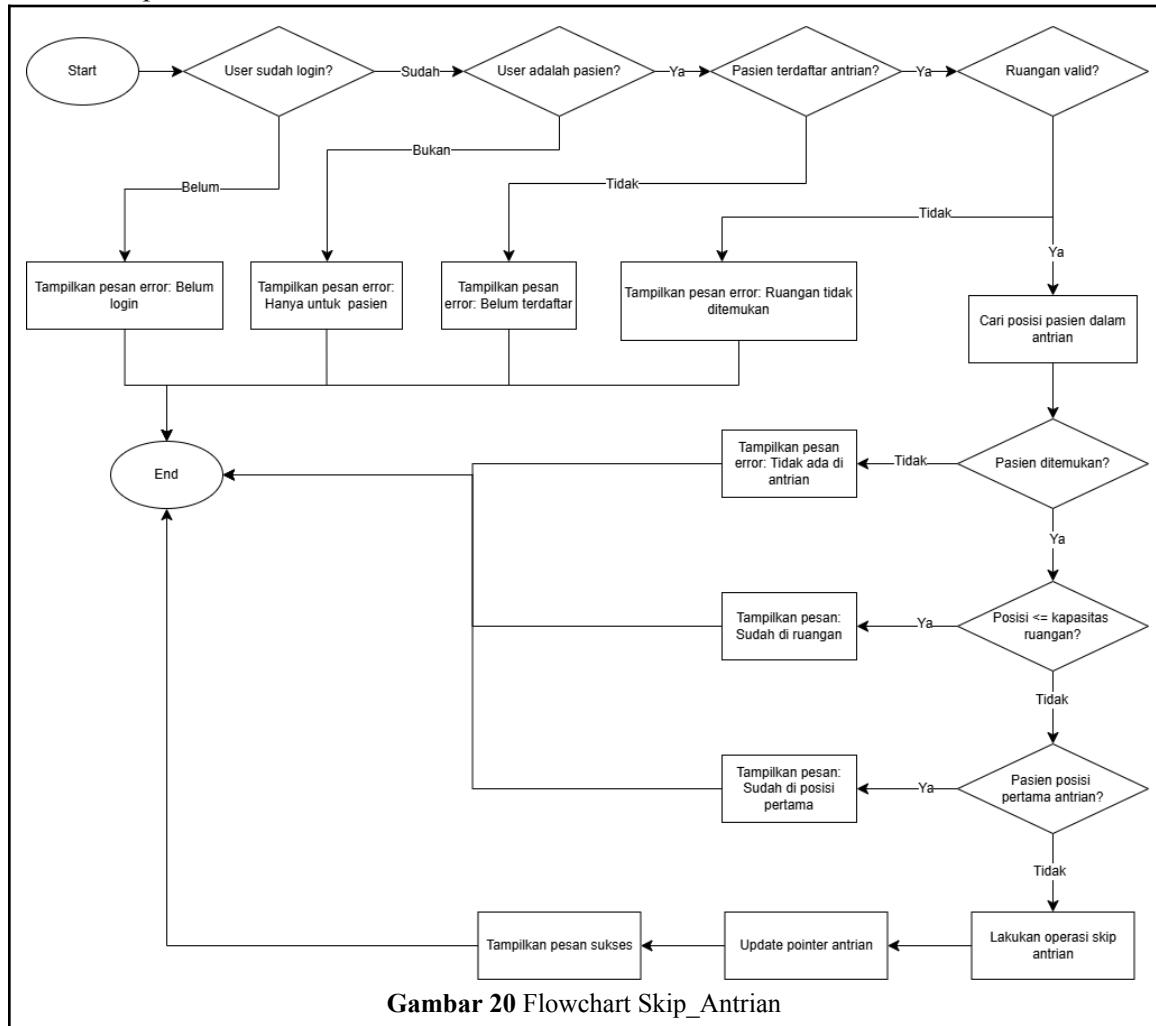


18. F18 - Exit



19. B06 - Mainin Antrian

- Skip Antrian



H. SPESIFIKASI FUNGSIONAL PROGRAM

1. F01 - Login

```
procedure login_system()
{login dengan memasukkan username dan password dan memvalidasinya}
I.S. Tidak ada user yang login. User memasukkan username dan password
F.S. User berhasil login atau user tidak dapat login

KAMUS LOKAL
Login, found_user, valid_password : Boolean
username, password, role : String
i : Integer

procedure login_system ()
    global login
    global current_user
    global username
    global role
    global users
    global user_count
    if login then
        output("Login gagal!")
        output('Anda telah login dengan username', username, 'silahkan
        lakukan "logout" sebelum melakukan login kembali.')
    else
        user <- input("Username : ")
        password <- input(Password : ")

        user_match <- False
        password_match <- False

        i<- 0
        while (i < user_count) do
            if (users[i].username == username) then
                user_match <- True
                if (users[i].password == password) then
                    password_match <- True
                    current_user <- &users[i]
                    break
            i ← i + 1

            if (user_match) then
                if (password_match) then
                    login <- True
```

```

username <- current_user.username
role <- current_user.role
output("Selamat pagi ", role_to_string(role), " ",
username, "!")
    if (role == "pasien") then
        output("Ada keluhan apa?")
    else
        output("")
    else
        output("Password salah untuk pengguna yang bernama ",
username, "!")
    else
        output("Tidak ada Manager, Dokter, atau pun Pasien yang
bernama ", username, "!")

```

2. F02 - Register

```

procedure register_pasien()
{Melakukan pendaftaran pasien baru ke dalam sistem}
I.S. Program aktif tanpa user yang login; user ingin mendaftar
sebagai pasien dengan username dan password.
F.S. Pasien terdaftar jika username valid dan belum dipakai, serta
tidak ada user yang login. Jika tidak, registrasi dibatalkan dengan
pesan kesalahan.

KAMUS LOKAL
is_valid_input, is_alpha_only, is_unique: Boolean
username, password : String

global current_user
global user_count
global users

if current_user != NULL then
    output("Anda sudah login. Silakan logout terlebih dahulu.")
else
    username <- input("Username:")
    if not is_alpha_string(username) then
        output("Username hanya boleh berisi huruf!")
        return

    if not is_username_unique(username) then
        output("Registrasi gagal! Pasien dengan nama ", username,
sudah terdaftar.)
        return

    password <- input("Password: ")

```

```

if user_count >= MAX_USERS then
    output("Kapasitas user penuh!")
    return

users[user_count].username <- username
users[user_count].password <- password
users[user_count].role <- ROLE_PASIEN
user_count <- user_count + 1

output("Pasien ", username, " berhasil ditambahkan!")

```

3. F03 - Logout

procedure F03() {Prosedur untuk melakukan logout dari sistem dan menghapus informasi user yang sedang login} I.S. Program dalam keadaan berjalan, dan variabel current_user menunjuk ke user yang sedang login (jika ada) F.S. Jika ada user yang login, maka user berhasil logout dan current_user diset menjadi NULL; jika belum login, sistem menampilkan pesan untuk login terlebih dahulu.
--

KAMUS LOKAL

is_logged_in : Boolean

global current_user if current_user == NULL then output ("Logout gagal!") output ("Anda belum login, silahkan login terlebih dahulu sebelum melakukan logout") else output ("Sampai jumpa ", current_user.username, "!") current_user <- Null
--

4. F04 - Lupa Password

procedure lupa_password() {Prosedur untuk mereset password user yang lupa dengan memverifikasi username dan kode unik} I.S. Program dalam keadaan berjalan, dan variabel current_user menunjuk ke user yang sedang login (jika ada) F.S. Jika username ditemukan dan kode unik sesuai, user diminta memasukkan password baru; jika tidak, sistem menampilkan pesan kesalahan yang sesuai.
--

KAMUS LOKAL

username, kode_unik_input, kode_unik_asli, password_baru : String
i : integer

```

Is_valid_input, is_kode_sesuai : Boolean

global users
global user_count
output ("Username: ")
input(username)
user <- NULL
for i <- 0 to user_count - 1 do
    if (users[i].username == username) then
        user <- &users[i]
        break
    if user == NULL then
        output("Username tidak terdaftar!")
        return
    output ("Kode Unik")
    input(kode_unik_input)

generate_kode_unik(user.username, kode_unik_asli)
if (kode_unik_input != kode_unik_asli) then
    output("Kode unik salah!")
    return

output("Halo ", role_to_string(user.role), " ", user.username, ",",
silakan daftarkan ulang password anda!")
output("Password Baru: ")
input(password_baru)

user.password <- password_baru
output("Password berhasil diubah!")

```

5. F05 - Menu & Help

```

procedure help_system()
{menampilkan daftar perintah yang tersedia sesuai status login dan
peran pengguna}
I.S. Pengguna dapat belum login atau sudah login sebagai Manager,
Dokter, atau Pasien
F.S. Sistem menampilkan perintah yang sesuai dengan status dan peran
pengguna, atau instruksi login/register jika belum login

```

KAMUS LOKAL

username : String

```

procedure help_system()
if current_user == NULL then
    output("Kamu belum login sebagai role apapun. Silahkan login
terlebih dahulu.\n")
    output("LOGIN: Masuk ke dalam akun yang sudah terdaftar")

```

```

output("REGISTER: Membuat akun baru")
else
    output("Halo ", role_to_string(current_user.role), " ",
current_user.username, ". ")

    if current_user.role == ROLE_DOKTER then
        output("Kamu memanggil command HELP. Kamu pasti sedang
kebingungan.")
        output("Berikut adalah hal-hal yang dapat kamu lakukan
sekarang:\n")
        output("LOGOUT: Keluar dari akun yang sedang digunakan")
        output("DIAGNOSIS: Melakukan diagnosis penyakit pasien
berdasarkan kondisi tubuh pasien")

    else if current_user.role == ROLE_PASIEN then
        output("Kamu memanggil command HELP. Kamu pasti sedang
kebingungan.")
        output("Berikut adalah hal-hal yang dapat kamu lakukan
sekarang:\n")
        output("LOGOUT: Keluar dari akun yang sedang digunakan")
        output("DAFTAR_CHECKUP: Mendaftarkan diri untuk pemeriksaan
dokter")

    else if current_user.role == ROLE_MANAGER then
        output("Kenapa kamu memanggil command HELP? Kan kamu manager,
tapi yasudahlah kamu pasti sedang kebingungan.")
        output("Berikut adalah hal-hal yang dapat kamu lakukan
sekarang:\n")
        output("LOGOUT: Keluar dari akun yang sedang digunakan")
        output("TAMBAH_DOKTER: Mendaftarkan dokter baru ke sistem")
output("\\nFootnote:")
output("Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang
terdaftar")
output("Jangan lupa untuk memasukkan input yang valid")

```

6. F06 - Denah Rumah Sakit

- Lihat denah

```

procedure lihat_denah(input L: ListRuang)
{Prosedur untuk menampilkan denah rumah sakit berupa nomor-nomor
ruangan}
I.S. Data ruangan sudah terisi dengan jumlah ruangan dan nomor
masing-masing.
F.S. Menampilkan daftar nomor ruangan dalam format denah horizontal.

```

KAMUS LOKAL

i : integer

```

global ruangan : ListRuangan

procedure lihat_denah(input: ListRuangan)

output_newline
output("-----+-----+-----+-----+-----+-----+")
i traversal[0... ruangan.jumlah - 1]
    output("| ", ruangan.ruang[i].nomor, " ", end="")
output("| ")
output("-----+-----+-----+-----+-----+-----+")

```

- Lihat Ruangan

```

procedure lihat_ruangan (input/output ruangan: ListRuangan, input num:
integer, input/output users: ListUser)

{Prosedur untuk menampilkan detail sebuah ruangan rumah sakit
berdasarkan nomor ruangan}
I.S. Pengguna sudah login dengan role Manager, Dokter, atau Pasien.
F.S. Jika nomor ruangan valid, detail ruangan ditampilkan; jika
tidak, tampilkan pesan kesalahan.

```

KAMUS LOKAL

```

r : pointer to Ruangan
i : integer
global ruangan : ListRuangan
global current_user : pointer to User

```

```

procedure lihat_ruangan (input: ListRuangan, input: integer)

If (current_user.role != ROLE_MANAGER and current_user.role !=
ROLE_DOKTER and current_user.role != ROLE_PASIEN) then
    output("Role tidak ditemukan. Akses ditolak.")
    ->

If (num <= 0 or num > ruangan.jumlah) then
    output("Nomor ruangan tidak valid.")
    ->

r <- &ruangan.ruang[num - 1]

output("--- Detail Ruangan ", r.nomor, " ---")
output("Kapasitas : ", r.kapasitas)
output("Dokter : ", r.dokter)
output("Pasien di dalam ruangan:")

```

```

if (r.pasien.Count == 0) then
    output("Tidak ada pasien di dalam ruangan saat ini.")
else then
    i traversal [0... r.pasien.Count - 1]
        output(" ", i + 1, ". ", r.pasien.Elements[i])
output("-----")

```

7. F07 - Lihat User

- To lower

FUNCTION to_lower (input c: char) -> char { IS: Sebuah karakter alfabet diberikan. } { FS: Jika karakter huruf kapital, dikonversi menjadi huruf kecil. Jika bukan huruf kapital, dikembalikan apa adanya. }
--

KAMUS LOKAL

ALGORITMA

```

if (c ≥ 'A' and c ≤ 'Z') then
    -> (c - 'A' + 'a')
else
    -> c

```

- Compare nama

FUNCTION compare_nama (input/output a: char, input/output b: char) -> integer integer { IS: Dua string nama diberikan dalam huruf campuran kapital dan kecil. } { FS: Mengembalikan -1 jika a < b, 0 jika a == b, dan 1 jika a > b, dibandingkan secara tidak case-sensitive. }
--

KAMUS LOKAL

i: integer
ca: karakter
cb: karakter

ALGORITMA

```

i ← 0
while (a[i] != NULL and b[i] != NULL) do
    ca ← to_lower(a[i])
    cb ← to_lower(b[i])
    if (ca != cb) then
        if (ca < cb) then

```

```

        -> -1
else
    -> 1
    i ← i + 1

if (a[i] == b[i]) then
    -> 0
else if (a[i] != NULL) then
    -> 1
else
    -> -1

```

- Sort user

```

PROCEDURE sort_user (input: L: ListUser, input idx: array of integer,
input len: integer, input id: integer, input asc: integer)
{ IS: Array indeks `idx` sudah terisi dengan indeks user yang ingin
diurutkan. }
{ FS: Indeks user dalam `idx` diurutkan berdasarkan ID jika `id = 1`,
atau berdasarkan nama jika `id = 2 (false/0)`, secara ascending jika
`asc = 1`, atau descending jika `asc = 2 (false/0)`. }

```

KAMUS LOKAL

i, j: integer
 cmp: integer
 tmp: integer

ALGORITMA

```

    i traversal [0... len - 2]
        j traversal [0... len - i - 2]
            if (id == 1) then
                cmp ← L.data[idx[j]].idUser - L.data[idx[j+1]].idUser
            else
                cmp ← compare_nama(L.data[idx[j]].username,
L.data[idx[j+1]].username)
            endif

            if (asc == 1 and cmp > 0) or (asc == 0 and cmp < 0) then
                tmp ← idx[j]
                idx[j] ← idx[j+1]
                idx[j+1] ← tmp
            endif
        endfor
    endfor

```

- Print user

```
PROCEDURE print_user (input: L: ListUser, idx: array of integer, len: integer, view: integer)
{ IS: Program dalam keadaan berjalan dan pengguna telah login. Data pengguna sudah tersedia dalam `L` dan sudah disortir dalam `idx`. }
{ FS: Menampilkan daftar user ke layar sesuai mode tampilan `view`: 0 = semua user, 1 = hanya pasien, 2 = hanya dokter. Data manajer tidak ditampilkan. }
```

KAMUS LOKAL

i, id: integer
 role_s: string
 penyakit: string

ALGORITMA

```
if (view == 0) then
    output("ID | Nama | Role | Penyakit")
    output("-----")
    i traversal [0... len - 1]
        id ← idx[i]
        if L.data[id].role == ROLE_MANAGER then
            continue
        endif
        role_s ← role_to_string(L.data[id].role)
        if (L.data[id].role == ROLE_PASIEN and
L.data[id].dataPasien != NULL) then
            penyakit ← L.data[id].dataPasien.penyakit
        else
            penyakit ← "-"
        endif
        output(L.data[id].idUser, " | ", L.data[id].username, " |
", role_s, " | ", penyakit)
    endfor

    else_if (view == 1) then
        output("ID | Nama | Penyakit")
        output("-----")
        i traversal [0... len - 1]
            id ← idx[i]
            if (L.data[id].role == ROLE_MANAGER) then
                continue
            endif
            if (L.data[id].role == ROLE_PASIEN) then
                if (L.data[id].dataPasien != NULL) then
                    penyakit ← L.data[id].dataPasien.penyakit
                else
                    penyakit ← "-"
                endif
```

```

        output(L.data[id].idUser, " | ", L.data[id].username,
" | ", penyakit)
        endif
    endfor

else if (view == 2) then
    output("ID | Nama")
    output("-----")
    i traversal [0... len - 1]
    id ← idx[i]
    if (L.data[id].role == ROLE_MANAGER) then
        continue
    endif
    if (L.data[id].role == ROLE_DOKTER) then
        output(L.data[id].idUser, " | ", L.data[id].username)
    endif
endfor
endif

```

- Lihat user

PROCEDURE lihat_user (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna telah login. }
{ FS: Jika pengguna merupakan Manager, sistem akan menampilkan seluruh
data pengguna (selain Manager), disortir berdasarkan ID atau Nama
secara ascending atau descending sesuai pilihan. Jika bukan Manager,
ditampilkan pesan akses ditolak. Jika belum login, ditampilkan pesan
bahwa pengguna belum login. }

KAMUS LOKAL

```

idx      : array [0..MAX_CAPACITY-1] of integer
len      : integer
pilihan1 : integer { 1 = ID, 2 = Nama }
pilihan2 : integer { 1 = ASC, 2 = DESC }
i        : integer
asc      : boolean
sortById : boolean

```

ALGORITMA

```

if (currUser == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    ->
endif

if (currUser.role != ROLE_MANAGER) then
    output("ERROR: Hanya manajer yang bisa melihat user.")

```

```

    ->
endif

output("Urutkan berdasarkan?\n1. ID\n2. Nama")
repeat
    input(">>> Pilihan: ", pilihan1)
until(pilihan1 == 1 or pilihan1 == 2)

output("Urutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)")
repeat
    input(">>> Pilihan: ", pilihan2)
until(pilihan2 == 1 or pilihan2 == 2)

len ← L.length
i traversal [0... len-1]
    idx[i] ← i
endfor

asc ← (pilihan2 == 1)
sortById ← (pilihan1 == 1)
sort_user(L, idx, len, sortById, asc)

output("Menampilkan semua pengguna dengan ",
        (sortById ? "ID" : "nama"),
        " terurut",
        (asc ? "ascending" : "descending"), "...")

output("+-----+")
output("| DAFTAR USER |")
output("+-----+")

print_user(L, idx, len, 0)

```

- Lihat pasien

PROCEDURE lihat_pasien (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna telah login. }
{ FS: Jika pengguna merupakan Manager, sistem akan menampilkan seluruh
data pasien, disortir berdasarkan ID atau Nama secara ascending atau
descending sesuai pilihan pengguna. Jika bukan Manager, ditampilkan pesan bahwa
pengguna belum login. }

KAMUS LOKAL

idx: array [0..MAX_CAPACITY-1] of integer
len: integer
pilihan1: integer

```
pilihan2: integer
i: integer
asc: boolean
sortById: boolean
```

ALGORITMA

```
if (currUser == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    ->
endif

if (currUser.role != ROLE_MANAGER) then
    output("ERROR: Hanya manajer yang bisa melihat pasien.")
    ->
endif

output("Urutkan berdasarkan?\n1. ID\n2. Nama")

repeat
    input(">>> Pilihan: ", pilihan1)
until (pilihan1 == 1 or pilihan1 == 2)

output("Urutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)")

repeat
    input(">>> Pilihan: ", pilihan2)
until (pilihan2 == 1 or pilihan2 == 2)

len ← L.length
i traversal [0... len-1]
    idx[i] ← i
endfor

asc ← (pilihan2 == 1)
sortById ← (pilihan1 == 1)
sort_user(L, idx, len, sortById, asc)

output("Menampilkan pasien dengan ",
       (sortById ? "ID" : "nama"),
       " terurut ",
       (asc ? "ascending" : "descending"), "...")

output("+-----+")
output("| DAFTAR PASIEN |")
output("+-----+")

print_user(L, idx, len, 1)
```

- Lihat dokter

```

PROCEDURE lihat_pasien (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna telah login. }
{ FS: Jika pengguna merupakan Manager, sistem akan menampilkan seluruh
data dokter, disortir berdasarkan ID atau Nama secara ascending atau
descending sesuai pilihan pengguna. Jika bukan Manager, ditampilkan pesan "Akses ditolak". Jika belum login, ditampilkan pesan bahwa
pengguna belum login. }

```

KAMUS LOKAL

```

idx      : array [0..MAX_CAPACITY-1] of integer
len      : integer
pilihan1 : integer
pilihan2 : integer
i        : integer
asc      : boolean
sortById : boolean

```

ALGORITMA

```

if (currUser == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    return
endif

if (currUser.role != ROLE_MANAGER) then
    output("ERROR: Hanya manajer yang bisa melihat dokter.")
    ->
endif

output("Urutkan berdasarkan?\n1. ID\n2. Nama")
repeat
    input(">>> Pilihan: ", pilihan1)
until(pilihan1 == 1 or pilihan1 == 2)

output("Urutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)")
repeat
    input(">>> Pilihan: ", pilihan2)
until(pilihan2 == 1 or pilihan2 == 2)

len ← L.length
i traversal [0... len-1]
    idx[i] ← i
endfor

asc ← (pilihan2 == 1)
sortById ← (pilihan1 == 1)

```

```

sort_user(L, idx, len, sortById, asc)

output("Menampilkan dokter dengan ",
         (sortById ? "ID" : "nama"),
         " terurut ",
         (asc ? "ascending" : "descending"), "...")

output("+-----+")
output("| DAFTAR DOKTER |")
output("+-----+")

print_user(L, idx, len, 2)

```

8. F08 - Cari User

- Binary search

FUNCTION binary_search (input: L: ListUser, input: targetElement: integer, input size: integer) -> **integer**
{ IS: Data `L` telah disortir berdasarkan ID. `targetElement` berisi ID yang ingin dicari. }
{ FS: Mengembalikan indeks dari user dengan ID yang cocok jika ditemukan, atau -1 jika tidak ditemukan. }

KAMUS LOKAL

left : integer
right : integer
middle : integer

ALGORITMA

```

left ← 0
right ← size - 1

while (left ≤ right) do
    middle ← (left + right) div 2

    if (L.data[middle].idUser == targetElement) then
        -> middle
    else if (L.data[middle].idUser < targetElement) then
        left ← middle + 1
    else
        right ← middle - 1
    endif
endwhile

-> -1

```

- sequential search username

```
FUNCTION sequential_username (input L: ListUser, input/output
targetName: char, input size: integer) -> integer
{ IS: Data `L` berisi daftar user. `targetName` adalah nama user yang
ingin dicari. }
{ FS: Mengembalikan indeks dari user dengan username yang cocok jika
ditemukan, atau -1 jika tidak ditemukan. }
```

KAMUS LOKAL

i : integer

ALGORITMA

```
i traversal [0... size - 1]
  if (L.data[i].username == targetName) then
    -> i
  endif
endfor

-> -1
```

- Search penyakit

```
PROCEDURE search_penyakit (input: L: ListUser, input/output
targetPenyakit: char)
{ IS: Data `L` sudah berisi daftar user lengkap, termasuk informasi
penyakit untuk pasien. }
{ FS: Menampilkan daftar pasien dengan penyakit yang cocok dengan
`targetPenyakit`. Jika tidak ada pasien ditemukan, ditampilkan pesan
bahwa tidak ditemukan. }
```

KAMUS LOKAL

i : integer
found : integer

ALGORITMA

```
found ← 0

output("ID | Nama | Penyakit")
output("-----")

i traversal [0... L.length - 1]
  if (L.data[i].role == ROLE_PASIEN) and (L.data[i].dataPasien ≠
NULL) then
    if (L.data[i].dataPasien.penyakit == targetPenyakit) then
      output(L.data[i].idUser, L.data[i].username,
L.data[i].dataPasien.penyakit)
```

```

        found ← found + 1
    endif
endif
endfor

if (found == 0) then
    output("Tidak ditemukan pasien dengan penyakit \",
targetPenyakit, "\".")
endif

```

- Cari user

PROCEDURE cari_user (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login. }
{ FS: Jika pengguna adalah manajer dan data user ditemukan berdasarkan ID atau username, maka informasi user akan ditampilkan (selain manajer). Jika pengguna belum login atau bukan manajer, akan ditampilkan pesan yang sesuai dan pencarian tidak dilanjutkan. Jika ID/nama tidak ditemukan, akan ditampilkan pesan bahwa user tidak ditemukan. }

KAMUS LOKAL

pilihan : integer
idTarget : integer
userIdx : integer
len : integer ← L.length
usernameTarget : string
u : User

ALGORITMA

```

if (currUser = NULL) then
    output("ERROR: Kamu belum login.")
    ->
endif

if (currUser.role ≠ ROLE_MANAGER) then
    output("ERROR: Hanya manajer yang bisa mencari user.")
    ->
endif

output("Cari berdasarkan?")
output("1. ID")
output("2. Nama")

repeat
    input(pilihan)

```

```

until_(pilihan = 1 or pilihan = 2)

if_(pilihan = 1) then
    input(idTarget)
    userIdx ← binary_search(L, idTarget, len)

    if_(userIdx = -1) then
        output("Tidak ditemukan pengguna dengan ID user ",
idTarget)
        ->
    endif
else
    input(usernameTarget)
    userIdx ← sequential_username(L, usernameTarget, len)

    if_(userIdx = -1) then
        output("Tidak ditemukan pengguna dengan nama ",
usernameTarget)
        ->
    endif
endif

u ← L.data[userIdx]

if_(u.role = ROLE_MANAGER) then
    output("ID/nama tersebut milik Manager dan tidak
ditampilkan.")
    ->
endif

role_s ← role_to_string(u.role)
penyakit ← jika u.role = ROLE_PASIEN dan u.dataPasien ≠ NULL maka
u.dataPasien.penyakit else "-"

output("ID | Nama | Role | Penyakit")
output("-----")
output(u.idUser, u.username, role_s, penyakit)

```

- Cari pasien

```

PROCEDURE cari_pasien (Input L: ListUser)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login. }
{ FS: Jika pengguna adalah manajer dan data pasien ditemukan
berdasarkan ID, nama, atau penyakit, maka informasi pasien akan
ditampilkan. Jika pengguna belum login atau bukan manajer, akan
ditampilkan pesan yang sesuai. Jika data tidak ditemukan atau user

```

```
bukan pasien, akan ditampilkan pesan yang sesuai. }
```

KAMUS LOKAL

```
pilihan      : integer
idTarget      : integer
userIdx       : integer
len           : integer ← L.length
usernameTarget : string
penyakitTarget : string
u             : User
role_s         : string
penyakit      : string
```

ALGORITMA

```
if (currUser = NULL) then
    output("ERROR: Kamu belum login.")
    ->
endif

if (currUser.role ≠ ROLE_MANAGER) then
    output("ERROR: Hanya manajer yang bisa mencari pasien.")
    ->
endif

output("Cari berdasarkan?")
output("1. ID")
output("2. Nama")
output("3. Penyakit")

repeat
    input(pilihan)
until (pilihan = 1 or pilihan = 2 or pilihan = 3)

if (pilihan = 1) then
    input(idTarget)
    userIdx ← binary_search(L, idTarget, len)

    if (userIdx = -1) then
        output("Tidak ditemukan pasien dengan ID ", idTarget)
        ->
    endif

    u ← L.data[userIdx]

    if (u.role = ROLE_MANAGER) then
        output("ID tersebut milik Manager dan tidak ditampilkan.")
        ->
```

```

else if (u.role ≠ ROLE_PASIEN) then
    output("User ditemukan, tetapi bukan seorang pasien.")
    ->
else
    penyakit ← jika u.dataPasien ≠ NULL maka
u.dataPasien.penyakit else "-"
    output("ID | Nama | Penyakit")
    output("-----")
    output(u.idUser, u.username, penyakit)
endif

else if (pilihan = 2) then
    input(usernameTarget)
    userIdx ← sequential_username(L, usernameTarget, len)

    if (userIdx = -1) then
        output("Tidak ditemukan pasien dengan nama ",
usernameTarget)
        ->
    endif

    u ← L.data[userIdx]

    if (u.role = ROLE_MANAGER) then
        output("Nama tersebut milik Manager dan tidak
ditampilkan.")
        ->
    else if (u.role ≠ ROLE_PASIEN) then
        output("User ditemukan, tetapi bukan seorang pasien.")
        ->
    else
        penyakit ← jika u.dataPasien ≠ NULL maka
u.dataPasien.penyakit else "-"
        output("ID | Nama | Penyakit")
        output("-----")
        output(u.idUser, u.username, penyakit)
    endif

else
    input(penyakitTarget)
    search_penyakit(L, penyakitTarget)
endif

```

- Cari dokter

PROCEDURE cari_dokter (Input L: ListUser) { IS: Program dalam keadaan berjalan. Pengguna sudah login. }

```
{ FS: Jika pengguna adalah manajer dan data dokter ditemukan berdasarkan ID atau nama, maka informasi dokter ditampilkan. Jika pengguna belum login atau bukan manajer, maka pencarian dibatalkan dengan menampilkan pesan yang sesuai. Jika user bukan dokter, maka ditampilkan pesan bahwa user bukan dokter. }
```

KAMUS LOKAL

```
pilihan      : integer
idTarget      : integer
userIdx       : integer
len           : integer ← L.length
usernameTarget : string
u             : User
```

ALGORITMA

```
if (currUser = NULL) then
    output("ERROR: Kamu belum login.")
    ->
endif

if (currUser.role ≠ ROLE_MANAGER) then
    output("ERROR: Hanya manajer yang bisa mencari dokter.")
    ->
endif

output("Cari berdasarkan?")
output("1. ID")
output("2. Nama")

repeat
    input(pilihan)
until (pilihan = 1 or pilihan = 2)

if (pilihan = 1) then
    input(idTarget)
    userIdx ← binary_search(L, idTarget, len)

    if (userIdx = -1) then
        output("Tidak ditemukan dokter dengan ID ", idTarget)
        ->
    endif
else
    input(usernameTarget)
    userIdx ← sequential_username(L, usernameTarget, len)

    if (userIdx = -1) then
        output("Tidak ditemukan dokter dengan nama ",
usernameTarget)
```

```

        ->
    endif
endif

u ← L.data[userIdx]

if (u.role = ROLE_MANAGER) then
    output("ID tersebut milik Manager dan tidak ditampilkan.")
    ->
else if (u.role ≠ ROLE_DOKTER) then
    output("User ditemukan, tetapi bukan seorang dokter.")
    ->
else
    output("ID | Nama")
    output("-----")
    output(u.idUser, u.username)
endif

```

9. F09 - Lihat Antrian

PROCEDURE lihat_semua_antrian(**Input/output** ruangan: ListRuang, **input/output** users: ListUser.)
{ IS: Program dalam keadaan berjalan. Pengguna sudah login dan ingin melihat semua antrian pasien di tiap ruangan. }
{ FS: Jika pengguna belum login, maka sistem menampilkan pesan untuk login terlebih dahulu dan fungsi dibatalkan.
- Jika pengguna bukan manajer, maka sistem menampilkan pesan akses ditolak dan fungsi dibatalkan.
- Jika pengguna adalah manajer, maka sistem menampilkan daftar nomor ruangan yang tersedia.
- Untuk setiap ruangan yang memiliki dokter, sistem menampilkan detail ruangan termasuk

KAMUS LOKAL

i	: integer
r	: Ruangan
countPasien	: integer
counter	: integer
current	: address

ALGORITMA

```

if (currUser == NULL) then
    output("Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.")
    ->
endif

```

```

if (currUser.role ≠ ROLE_MANAGER) then
    output("Anda bukan Manajer, akses ditolak")
    ->
endif

output("")
output("-----+")
output("|           SEMUA ANTRIAN           |")
output("-----+")

output("-----+-----+-----+-----+-----+-----+-----+-----+")
i traversal [0... ruangan.jumlah - 1]
    output("|   ", ruangan.ruang[i].nomor, "   ", end="")
endfor
output("|")
output("-----+-----+-----+-----+-----+-----+-----+-----+")
output("")

for i ← 0 to ruangan.jumlah - 1 do
    r ← ruangan.ruang[i]

    if (r.dokter = NULL or panjang(r.dokter.username) = 0) then
        continue
    endif

    output("--- Detail Ruangan ", r.nomor, " ---")
    output("Kapasitas : ", r.kapasitas)
    output("Dokter : Dr. ", r.dokter.username)
    output("Pasien di dalam ruangan:")

    countPasien ← 0
    current ← r.Antrian.first

    while (current ≠ NULL and countPasien < r.kapasitas) do
        countPasien ← countPasien + 1
        output(" ", countPasien, ". ", current.pasien.username)
        current ← current.next
    endwhile

    if (countPasien = 0) then
        output(" Tidak ada pasien di dalam ruangan saat ini.")
    endif

    output("Pasien di antrian:")
    counter ← 1

    while (current ≠ NULL) do

```

```

        output(" ", counter, ". ", current.pasien.username)
        counter ← counter + 1
        current ← current.next
    endwhile

    if (counter = 1) then
        output(" Tidak ada pasien di antrian saat ini.")
    endif

    output("")
endfor

```

10. F10 - Tambah Dokter

- Tambah dokter

PROCEDURE tambah_dokter (input/output: ListUser, input/output: Set)
{Prosedur untuk menambahkan user baru dengan role dokter, hanya bisa dilakukan oleh Manajer}
I.S. Sistem dalam keadaan berjalan dan pengguna sudah login.
F.S. Jika valid, dokter baru ditambahkan ke list pengguna dan username-nya disimpan di set; jika tidak, tampilkan pesan kesalahan.

KAMUS LOKAL

username, password : string
new_user : User
global current_user : pointer to User
global users : ListUser
global usernames : Set

PROCEDURE tambah_dokter (input/output: ListUser, input/output: Set)

If (current_user == NULL) then
 output("Tidak ada pengguna yang sedang login. Silakan login terlebih dahulu.")
 ->

if (current_user.role != ROLE_MANAGER) then
 output("Akses ditolak. Anda bukan Manager.")
 ->

output("Username: ")
input(username)

if (!is_alpha_string(username)) then
 output("Username hanya boleh berisi huruf!")
 ->

```

if (!is_username_unique(usernames, username)) then
    output("Sudah ada dokter bernama ", username, "!")
    ->

output("Password: ")
(password)

if (IsFull(users)) then
    output("Kapasitas user penuh!")
    ->

new_user.username <- username
new_user.password <- password
new_user.role <- ROLE_DOKTER

SetEl(users, NbElmt(users), new_user)
SetLength(users, NbElmt(users) + 1)
Insert(usernames, username)

output("Dokter ", username, " berhasil ditambahkan!")

```

- Assign dokter

PROCEDURE assign_dokter(: ListUser, : ListRuangan)
{Prosedur untuk menetapkan seorang dokter ke ruangan tertentu,
dilakukan oleh Manager}
I.S. Sistem dalam keadaan berjalan dan pengguna sudah login.
F.S. Jika valid, dokter ditugaskan ke ruangan; jika tidak, tampilkan
pesan kesalahan.

KAMUS LOKAL

username : string
nomor_ruangan, ruangan_idx : integer
dokter_baru : User
dokter_found, ruangan_found : boolean
global current_user : pointer to User
global users : ListUser
global ruangan : ListRuangan

PROCEDURE assign_dokter(: ListUser, : ListRuangan)

```

if (current_user == NULL) then
    output("Tidak ada pengguna yang sedang login. Silakan login
terlebih dahulu.")
    ->

```

```

if (current_user.role != ROLE_MANAGER) then
    output("Akses ditolak. Anda bukan Manager.")
    ->

output("Username: ")
input(username)

if (!is_alpha_string(username)) then
    output("Username hanya boleh berisi huruf!")
    ->

dokter_found <- false
i traversal [GetFirstIdx(users)... GetLastIdx(users)]
    if (users[i].username == username and users[i].role ==
ROLE_DOKTER) then
        dokter_baru <- users[i]
        dokter_found <- true
        break

if (!dokter_found) then
    output("Dokter dengan username ", username, " tidak ditemukan!")
    ->

output("Ruangan: ")
input(nomor_ruangan)

ruangan_found <- false
ruangan_idx <- -1
i traversal [0... ruangan.jumlah - 1]
    if (ruangan.ruang[i].nomor == nomor_ruangan) then
        ruangan_found <- true
        ruangan_idx <- i
        break

if (!ruangan_found) then
    output("Tidak ada ruangan nomor ", nomor_ruangan, "!")
    return

i traversal [0... ruangan.jumlah - 1]
    if (ruangan.ruang[i].dokter == username) then
        output("Dokter ", username, " sudah diassign ke ruangan ",
ruangan.ruang[i].nomor, "!")
        ->

if (ruangan.ruang[ruangan_idx].dokter == "") then
    ruangan.ruang[ruangan_idx].dokter <- username
    output("Dokter ", username, " berhasil diassign ke ruangan ",

```

```

nomor_ruangan, "!")
->

else_if (ruangan.ruang[ruangan_idx].dokter != "-" and ruangan_idx != -1) then
    output("Dokter ", ruangan.ruang[ruangan_idx].dokter, " sudah menempati ruangan ", nomor_ruangan, "!")
    output("Silakan cari ruangan lain untuk dokter ", username, ".")
->

```

11. F11 - Diagnosis

- Identifikasi Penyakit

FUNCTION identifikasi_penyakit (input/output pasien: Pasien) → **static character**

{ IS: Data pasien (suhu, tekanan darah, dll.) sudah terisi.
{ FS: Mengembalikan nama penyakit jika gejala pasien cocok dengan kriteria penyakit tertentu, atau NULL jika tidak ditemukan.}

KAMUS LOKAL

p : Penyakit

ALGORITMA

```

static namaPenyakit[MAX_PENYAKIT]
i traversal [0..jumlahPenyakit-1]
    Penyakit p ← ketPenyakit[i]
    if (pasien.suhu >= p.suhuMin) and (pasien.suhu <= p.suhuMax)
and
    (pasien.tekananDarah[0] >= p.bpSisMin) and
(pasien.tekananDarah[0] <= p.bpSisMax) and
    (pasien.tekananDarah[1] >= p.bpDiasMin) and
(pasien.tekananDarah[1] <= p.bpDiasMax) and
    (pasien.detakJantung >= p.bpmMin) and (pasien.detakJantung
<= p.bpmMax) and
    (pasien.saturasiOksigen >= p.satsMin) and
(pasien.saturasiOksigen <= p.satsMax) and
    (pasien.kadarGulaDarah >= p.bgMin) and
(pasien.kadarGulaDarah <= p.bgMax) and
    (pasien.beratBadan >= p.bbMin) and (pasien.beratBadan <=
p.bbMax) and
    (pasien.tinggiBadan >= p.tbMin) and (pasien.tinggiBadan <=
p.tbMax) and
    (pasien.kadarKolesterol >= p.fatMin) and
(pasien.kadarKolesterol <= p.fatMax) and
    (pasien.trombosit >= p.tromMin) and (pasien.trombosit <=
p.tromMax) then
        copy_string(pasien.penyakit, p.nama)
        copy_string(namaPenyakit, p.nama)

```

```

        → namaPenyakit
    endif
→ NULL

```

- Diagnosis Pasien

```

PROCEDURE diagnosis_pasien (input/output userDokter: User)
{ IS: Program dalam keadaan berjalan, user dokter sedang login, dan
memiliki pasien dalam antrian ruangan.}
{ FS: Bergantung pada kondisi pasien, sistem akan:
- Mengubah status pasien (butuhDiberiObat atau butuhPulang).
- Menampilkan pesan sesuai hasil diagnosis.
- Memastikan data pasien terupdate di sistem.
}

```

KAMUS LOKAL

```

i : integer
dokter : Dokter
r : Ruangan
current : address
userPasien : User
pasien : Pasien
penyakit : character

```

ALGORITMA

```

if (userDokter == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    return
endif

if (userDokter.role != ROLE_DOKTER) then
    output("ERROR: Hanya dokter yang bisa diagnosis!")
    return
endif

*dokter ← userDokter.dataDokter
idxRuang ← dokter->nomorRuang - 1

i traversal [0..ruangan.jumlah-1]
    if (ruangan.ruang[i].dokter != NULL) and
(ruangan.ruang[i].dokter.id == dokter.id) then
        idxRuang ← i
        break
    endif

if (idxRuang < 0) or (idxRuang >= ruangan.jumlah) then
    output("[dr. ", userDokter.username, "] ", "Kamu belum

```

```

memiliki ruangan.")
    return
endif

*r ← &ruangan.ruang[idxRuang]
current ← r->Antrian.first

if (current == NULL) then
    output("[dr. ", userDokter.username, "] ", "Kamu lagi nggak
ada pasien. Asik, free time!")
    return
endif

*userPasien ← current.pasien
*pasien ← userPasien.dataPasien

if (pasien.status == butuhDiberiObat) then
    output("[dr. ", userDokter.username, "] ", "Pasien ",
userPasien->username, "udah didiagnosa, tinggal dikasih obat aja.")
    return
endif

if (pasien->status != butuhDiagnosa) then
    output("[dr. ", userDokter.username, "] ", "Pasien ",
userPasien->username, "tidak perlu didiagnosis lagi.")
    return
endif

*penyakit = identifikasi_penyakit(pasien)
output("-----+")
output("|          DIAGNOSA          |")
output("-----+")
if (penyakit != NULL) then
    pasien->status ← butuhDiberiObat
    printf("[dr. ", userDokter.username, "] ", "Pasien @",
userPasien->username, " terdiagnosa mengidap penyakit: ", penyakit)
    output("[dr. ", userDokter.username, "] ", "Jangan lupa untuk
diobatin ya!")
    output("HELP: Untuk mengobati pasien @", userPasien.username,
" ketik NGOBATIN!")
else
    pasien.status ← butuhPulang
    output("[dr. ", userDokter.username, "] ", "Pasien @",
userPasien->username, " sehat banget! Dijamin kuat salto keliling
kota!")
endif

```

- Identifikasi Penyakit

```
FUNCTION cari_obat (input/output namaPenyakit: constant character) →  
PenyakitObatEntry  
{ IS: Nama penyakit valid dan database penyakitObatMap tersedia.}  
{ FS:  
    - Jika penyakit ditemukan: Mengembalikan pointer ke  
    PenyakitObatEntry yang berisi daftar obat.  
    - Jika tidak ditemukan: Mengembalikan NULL.  
}
```

KAMUS LOKAL**ALGORITMA**

```
i traversal [0..jumlahPenyakit-1]  
if (compare_string(penyakitObatMap[i].namaPenyakit,  
namaPenyakit) == 0) then  
    → &penyakitObatMap[i]  
endif  
→ NULL
```

- Ngobatin Pasien

```
PROCEDURE ngobatin (input/output currUser: User, input/output users:  
User, input banyakUser: integer, input/output ruangan: ListRuangan)  
{ IS: User dokter sedang login dan user adalah dokter. Data ruangan  
dan pasien dalam antrian tersedia.}  
{ FS:  
    - Jika valid, status pasien diubah menjadi butuhMinumObat dan  
    daftar obat disimpan.  
    - Pesan sesuai kondisi ditampilkan.  
}
```

KAMUS LOKAL

```
dokter : Dokter  
ruanganDokter : Ruangan  
i : integer  
current : address  
pasien : Pasien  
userPasien : User
```

ALGORITMA

```
if (currUser == NULL) then  
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu  
dengan command LOGIN.")  
    return  
endif
```

```

if (currUser.role != ROLE_DOKTER) then
    output("ERROR: Hanya dokter yang punya kemampuan mengobati
pasien!")
    return
endif

*dokter ← currUser.dataDokter
if (dokter == NULL) then
    output("ERROR: Data dokter tidak valid!")
    return
endif

*ruanganDokter ← NULL
i traversal [0..ruangan.jumlah-1]
if (ruangan.ruang[i].dokter != NULL) and
(ruangan.ruang[i].dokter.id == dokter.id) then
    ruanganDokter ← &ruangan.ruang[i]
    break
endif

output("-----+")
output("|           NGOBATIN           |")
output("-----+")

if (ruanganDokter == NULL) then
    output("[dr. ", currUser.username, "] ", "Kamu belum memiliki
ruangan!")
    return
endif

current ← ruanganDokter->Antrian.first
if (current == NULL) or (current.pasien.dataPasien == NULL) then
    output("[dr. ", currUser.username, "] ", "Kamu lagi nggak ada
pasien. Ngobatin siapa?")
    return
endif

*pasien ← current.pasien.dataPasien
*userPasien ← current.pasien

if (pasien.status == butuhDiagnosa) then
    output("[dr. ", currUser.username, "] ", "Pasien @",
userPasien.username, " belum didiagnosis. Diagnosis dulu sebelum
mengobati!")
    return
endif

```

```

        if (pasien.status == butuhPulang) then
            output("[dr. ", currUser.username, "] ", "Pasien @",
userPasien.username, " sudah sembuh, tidak perlu diobati lagi.")
            return
        endif

        if (pasien.status == butuhMinumObat) then
            output("[dr. ", currUser.username, "] ", "Pasien @",
userPasien.username, " sudah diberi obat, tidak perlu diobati lagi.")
            return
        endif

        *entry ← cari_obat(pasien.penyakit)
        if (entry == NULL) then
            output("[dr. ", currUser.username, "] ", "Tidak ada obat untuk
penyakit ", pasien.penyakit, "!")
            return
        endif

        output("[dr. ", currUser.username, "] ", "Resep untuk @",
userPasien.username, " (Penyakit: ", pasien.penyakit, "):")
        i traversal [0..entry.jumlahObat-1]
        output("      ", i+1, ". ", entry.obat[i].nama)

        pasien.jumlahObat ← entry.jumlahObat
        pasien.jumlahObatResep ← entry.jumlahObat

        i traversal [0..entry.jumlahObat-1]
        pasien.daftarObat[i] ← entry.obat[i]
        pasien.daftarObatResep[i] ← entry.obat[i]

        pasien.status ← butuhMinumObat

```

13. F13 - Aku boleh pulang ga, dok 😊?

```

procedure pulangdok (Input: Diagnosis: boolean, resep: List,
konsumsi_obat: Stack)
{ IS: User yang login pada program adalah pasien. Keadaan pasien
dalam sudah atau belum didiagnosis ditentukan oleh procedure
Diagnosis. Apabila sudah di diagnosis, terdapat urutan obat yang
diresepkan dokter dalam bentuk List, serta obat-obat yang sudah
diminum oleh pasien disusun dalam bentuk Stack}
{ FS: Program tidak memperbolehkan pasien pulang apabila:
- Belum menerima diagnosis dari dokter
- Ada obat yang belum dihabiskan
- Urutan obat yang diminum tidak sesuai resep dokter.
Apabila semua kondisi telah dipenuhi, program menampilkan pesan yang

```

```
memperbolehkan pasien untuk pulang.)
```

KAMUS LOKAL

```
userPasien : User
global ruangan : ListRuang
pasien : Pasien
ruangan Dokter : Ruangan
daftarObat : Obat
i,jumlahObat : integer
perutPasien : Stack
```

ALGORITMA

```
if (currUser == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    return
if (currUser.role != ROLE_PASIEN) then
    output("ERROR: Hanya pasien yang bisa konsultasi pada dokter
lagi!")
    return

*pasien ← user_pasien.dataPasien
idRuang ← pasien.idRuang
*ruanganDokter ← NULL

i traversal [0..listRuang.jumlah-1]
if (listRuang.ruang[i].nomor == idRuang) then
    ruanganDokter ← &listRuang.ruang[i]
    break

if (ruanganDokter == NULL) then
    output("ERROR: Ruangan pasien tidak ditemukan!")
    return

*antrianDokterRuang ← &ruanganDokter.Antrian
if (antrianDokterRuang.first == NULL) or
(antrianDokterRuang.first.pasien != user_pasien) then
    output(">> Maaf, kamu bukan pasien paling depan di antrian.
Tunggu giliranmu ya.")
    return

if (pasien.status == butuhDiagnosa) then
    output("[@, user_pasien.username, "] Loh... kamu belum
didiagnosa dokter sama sekali, jangan buru-buru pulang!")
    output("HELP: Untuk daftar gunakan DAFTAR_CHECKUP dan untuk
cek antrianmu gunakan LIHAT_ANTRIAN!")
```

```

        return

        if (pasien.status == butuhMinumObat) and (pasien.jumlahObat > 0)
then
            output("Kamu belum menghabiskan seluruh obat yang diberikan.
Yuk minum obatmu dulu!")
            return

            if (pasien.status == butuhPenawar) then
                output("Dokter sedang memeriksa...")
                output("Maaf, kamu kayaknya masih belum bisa pulang, masih
sakit ya?")

                output("Urutan obat yang diharapkan:")
                output("           ", gabungkanNamaObat(pasien.daftarObatResep,
pasien.jumlahObatResep))

                output("Urutan obat yang kamu minum:")
                output("           ",
gabungkanNamaObat(pasien.perutPasien.buffer, pasien.perutPasien.idxTop
+ 1))

                output("Kunjungi dokter untuk meminta penawar yang sesuai
yaa!")
                return

                output("Dokter sedang memeriksamu...")
                output("Yayy Selamat! Kamu sudah dinyatakan sembuh oleh
dokter. Silahkan pulang dan semoga sehat selalu")

                call make_default_pasien(pasien)
                keluar ← queue_dequeue(antrianDokterRuang)
                return
endif

```

14. F14 - Daftar Check Up

- Validasi Float

```

FUNCTION validasi_float (input nilai: float, input/output namaVar:
constant character) → boolean
{ IS: Variabel nilai dan namaVar telah diberikan.}
{ FS:
    - Jika nilai positif: Mengembalikan true dan tidak menampilkan
pesan.
    - Jika nilai negatif: Menampilkan pesan error dan mengembalikan
false.
}

```

KAMUS LOKAL**ALGORITMA**

```
if (nilai <= 0) then
    output("ERROR: ", namaVar, " harus positif!")
    → false
endif
→ true
```

- Validasi Integer

```
FUNCTION validasi_integer (input nilai: integer, input/output namaVar:
constant character) → boolean
{ IS: Variabel nilai dan namaVar telah diberikan.}
{ FS:
    - Jika nilai positif: Mengembalikan true dan tidak menampilkan
      pesan.
    - Jika nilai negatif: Menampilkan pesan error dan mengembalikan
      false.
}
```

KAMUS LOKAL**ALGORITMA**

```
if (nilai <= 0) then
    output("ERROR: ", namaVar, " harus positif!")
    → false
endif
→ true
```

- Display Dokter

```
PROCEDURE display_dokter (input/output users: User, input banyakUser:
integer, input/output ruangan: ListRuangan)
{ IS: Array users dan struktur ruangan tersedia dan valid. Variabel
banyakUser berisi jumlah user yang valid.}
{ FS:
    - Menampilkan daftar dokter beserta informasi ruangan dan jumlah
      antrian.
    - Jika ditemukan data dokter tidak valid, menampilkan pesan error
      yang spesifik.
}
```

KAMUS LOKAL

```
banyakDokter, i, j : integer
dokter : Dokter
```

ruanganDokter : Ruangan

ALGORITMA

```
output("Berikut adalah daftar dokter yang tersedia:")
banyakDokter ← 0

i traversal [0..banyakUser-1]
    if (users[i].role != ROLE_DOKTER) then
        continue
    endif

    if (users[i].dataDokter == NULL) then
        output("ERROR: Data dokter tidak valid untuk dr. ",
users[i].username, "!")
        continue
    endif

    *dokter ← users[i].dataDokter
    banyakDokter ← banyakDokter + 1

    if (dokter.nomorRuangan == '\0') then
        output(banyakDokter, ". dr. ", users[i].username, " (Belum
memiliki ruangan)")
        continue
    endif

    *ruanganDokter ← NULL
    j traversal [0..ruangan.jumlah-1]
        if (ruangan.ruang[j].dokter != NULL) and
(ruangan.ruang[j].dokter.id == dokter.id) then
            ruanganDokter ← &ruangan.ruang[j]
            break
        endif

    output(banyakDokter, ". dr. ", users[i].username, " (Ruang
", dokter.nomorRuangan, ") - Total Ruangan + Antrian: ",
ruanganDokter.Antrian.jumlah)
```

- Daftar Check Up

```
PROCEDURE daftar_check_up (input/output currUser: User, input/output
users: User, input banyakUser: integer, input/output ruangan:
ListRuang)
{ IS: Program berjalan, user pasien sedang login. Data pasien valid.}
{ FS: Bergantung pada kondisi, sistem akan:
- Memasukkan pasien ke antrian dokter yang dipilih.
- Mengupdate posisiAntrian pasien.
```

```
- Menampilkan pesan sukses/gagal beserta detail antrian.  
}
```

KAMUS LOKAL

```
i, j, banyakDokter, pilihan, idxDokter : integer  
dokter, dokterPilihan : Dokter  
ruanganDokter, r : Ruangan  
pasien, result : Pasien  
curr : address  
userDokter : User
```

ALGORITMA

```
if (currUser == NULL) then  
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu  
dengan command LOGIN.")  
    return  
endif  
  
if (currUser.role != ROLE_PASIEN) then  
    output("ERROR: Hanya pasien yang bisa daftar check-up.")  
    return  
endif  
  
if (currUser.dataPasien == NULL) then  
    output("ERROR: Data pasien tidak valid!")  
    return  
endif  
  
*pasien ← currUser.dataPasien  
  
if (pasien.posisiAntrian >= 0) then  
    output("[@", currUser.username, "] ", "Kamu sudah terdaftar di  
antrian!")  
    output("    Silakan selesaikan check-up yang sudah terdaftar  
terlebih dahulu.")  
    return  
endif  
  
i traversal [0..ruangan.jumlah-1]  
*r ← &ruangan.ruang[i]  
current ← r.Antrian.first  
while (current != NULL) do  
    if (current.pasien.dataPasien == pasien) then  
        output("[@", currUser.username, "] ", "Kamu sudah  
berada di ruangan ", r.nomor, " (atau antrian ruangan ", r.nomor, ")")  
        return  
    endif  
    current ← current.next
```

```

endwhile

banyakDokter ← 0
i traversal [0..banyakUser-1]
    if (users[i].role == ROLE_DOKTER) then
        banyakDokter ← banyakDokter + 1
    endif

output("-----+")
output("| DAFTAR CHECK UP |")
output("-----+")

output(">> Input data medis:")
repeat
    output("Suhu tubuh (Celcius): ")
    input(pasien.suhu)
until (validasi_float(pasien.suhu, "Suhu"))

repeat
    output("Tekanan darah (sistol/diastol, contoh: 120 80): ")
    input(pasien.tekananDarah[0], pasien.tekananDarah[1])
until (pasien.tekananDarah[0] > 0 and pasien.tekananDarah[1] > 0)

repeat
    output("Detak jantung (bpm): ")
    input(pasien.detakJantung)
until (validasi_integer(pasien.detakJantung, "Detak jantung"))

repeat
    output("Saturasi oksigen (persentase): ")
    input(pasien.saturasiOksigen)
until (validasi_float(pasien.saturasiOksigen, "Saturasi oksigen"))

repeat
    output("Kadar gula darah (mg/dL): ")
    input(pasien.kadarGulaDarrah)
until (validasi_integer(pasien.kadarGulaDarrah, "Kadar gula
darah"))

repeat
    output("Berat badan (kg): ")
    input(pasien.beratBadan)
until (validasi_float(pasien.beratBadan, "Berat badan"))

repeat
    output("Tinggi badan (cm): ")

```

```

        input(pasien.tinggiBadan)
        until (validasi_integer(pasien.tinggiBadan, "Tinggi badan"))

repeat
    output("Kadar kolesterol (mg/dL): ")
    input(pasien.kadarKolesterol)
until (validasi_integer(pasien.kadarKolesterol, "Kadar
kolesterol"))

repeat
    output("Trombosit (ribu/ $\mu$ L): ")
    input(pasien.trombosit)
until (validasi_integer(pasien.trombosit, "Trombosit"))

display_dokter(users, banyakUser, ruangan)

repeat
    output("Pilih dokter (1-", banyakDokter, "): ")
    input(pilihan)
until (pilihan >= 1 and pilihan <= banyakDokter)

idxDokter ← 0
*dokterPilihan ← NULL
*userDokter ← NULL

i traversal [0..banyakUser-1]
if (users[i].role == ROLE_DOKTER) then
    idxDokter ← idxDokter + 1
    if (idxDokter == pilihan) then
        userDokter ← &users[i]
        dokterPilihan ← users[i].dataDokter
        break
    endif
endif

if (dokterPilihan == NULL) then
    output("ERROR: Dokter tidak ditemukan!")
    return
endif

*result ← assign_pasien_ke_dokter(currUser, dokterPilihan, pasien,
ruangan)
if (result != NULL) then
    if (dokterPilihan.nomorRuangan == '\0') then
        output("ERROR: Dokter ", userDokter.username, " belum
memiliki ruangan. Mendaftarkan ke antrian...")
    else

```

```

        output(">> Pasien berhasil didaftarkan ke ruangan",
dokterPilihan.nomorRuangan, ".")
        endif

        if (result.posisiAntrian == -1) then
            output(">> Pasien berhasil didaftarkan ke antrian dokter
", userDokter.username, ".")
            pasien.posisiAntrian ← result.posisiAntrian
        else
            pasien.posisiAntrian ← result.posisiAntrian
        endif

        output(">> Pendaftaran berhasil! <<")
        output("[@", currUser.username, "] ", "Kamu terdaftar pada
antrian dr. ", userDokter.username, " di ruangan ",
dokterPilihan.nomorRuangan, ".")
        output("[@", currUser.username, "] ", "Posisi antrian Kamu: ",
result.posisiAntrian)
        else
            output("ERROR: Gagal mendaftar!")
        endif
    
```

15. F15 - Antrian Saya

```

PROCEDURE cek_antrian_saya (input/output user: User, input/output
users: User, input banyakUser: integer, input/output ruangan:
ListRuang)
{ IS: User pasien sedang login. Data pasien valid. Struktur ruangan
dan users tersedia.}
{ FS:
    - Menampilkan status antrian pasien atau pesan error sesuai
    kondisi
    - Tidak mengubah data sistem
}

```

KAMUS LOKAL

```

pasien : Pasien
found, posisi, idxRuang, i, totalAntrian : integer
curr : address
dokterPasien : Dokter

```

ALGORITMA

```

if (currUser == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    return
endif

```

```

if (user.role != ROLE_PASIEN) then
    output("ERROR: Hanya pasien yang bisa melihat antrian!")
    return
endif

*pasien ← user.dataPasien

found ← 0
posisi ← 0
idxRuang ← -1
i traversal [0..ruangan.jumlah-1]
    current ← ruangan.ruang[i].Antrian.first
    posisi ← 0
    while (current != NULL) do
        if (current.pasien.dataPasien == pasien) then
            found ← 1
            idxRuang ← i
            break
        endif
        current ← current.next
        posisi ← posisi + 1
    endwhile
    if (found) then
        break
    endif

if (not found) then
    if (pasien.posisiAntrian < 0) then
        output("[@", user.username, "] ", "Kamu nih belum
terdaftar dalam antrian check-up (berkata dengan nada lemah lembut
gemulai). Daftar dulu coba :D")
        output("HELP: Silakan daftar terlebih dahulu dengan
command DAFTAR_CHECKUP.")
    endif
    return
endif

if (posisi < MAX_PASIEN_RUANGAN) then
    output("[@", user.username, "] ", "Kamu lagi di ruangan
dokter, loh. Dokternya ga lagi tidur kan?")
    return
endif

if (idxRuang < 0) or (idxRuang >= ruangan.jumlah) then
    output("ERROR: Ruangan tidak ditemukan!")
    return
endif

```

```

if (ruangan.ruang[idxRuang].Antrian.first == NULL) then
    output("ERROR: Antrian ruangan ",
ruangan.ruang[idxRuang].nomor, " kosong!")
    return
endif

*ruanganDokter ← &ruangan.ruang[idxRuang]
*dokterPasien ← ruanganDokter.dokter

if (dokterPasien == NULL) then
    output("ERROR: Dokter pasien ", user.username, " tidak
ditemukan!")
    return
endif

totalAntrian ← queue_size(&ruangan.ruang[idxRuang].Antrian)

output("-----+")
output("|           LIHAT ANTRIAN           |")
output("-----+")

output("[@", user.username, " ] ", "Status antrian Anda:")
output("      Dokter: dr. ", dokterPasien.username)
output("      Ruangan: ", dokterPasien.nomorRuangan)
output("      Posisi antrian: ", posisi - MAX_PASIEN_RUANGAN + 1, "
dari ", totalAntrian - MAX_PASIEN_RUANGAN)

```

16. F16 - Minum Obat

procedure minumObat (**input/output**: currUser: User, **input/output**: Obat: daftarObat, **output**: Pasien: perutPasien)

{ IS: Inventory pasien berisikan daftar obat yang dimiliki oleh pasien pada saat ini, disusun dalam bentuk List. Serta obat-obat yang sudah diminum oleh pasien disusun dalam bentuk Stack}

{ FS: Jika nomor obat yang dipilih valid:

- Obat tersebut dihapus dari inventory
- Obat dimasukkan ke dalam Stack konsumsi
- Program menampilkan bahwa obat berhasil dimakan

Jika nomor obat yang dipilih tidak valid:

- Program menampilkan bahwa obat tidak tersedia
- Tidak terjadi perubahan pada inventory dan Stack konsumsi}

KAMUS LOKAL

currUser, userPasien : User
perutPasien : Stack
pasien : Pasien
pilihanObat, valid, idxObat, i : int

```
konfirmasi : character
```

```
ALGORITMA
```

```
    if (currUser == NULL) then
        output("Kamu belum login. Silakan login terlebih dahulu dengan
command LOGIN.")
        return
    endif

    if (currUser.role != ROLE_PASIEN) then
        output("Eits...pasien doang yang bisa minum obat!")
        return
    endif

*pasien ← user_pasien.dataPasien
*perutPasien ← &pasien.perutPasien

    if (pasien.status == butuhMinumObat or pasien.status ==
butuhPenawar) then
        output("Yuk minum obatmu dulu..")
        output("Kamu punya ", pasien.jumlahObat, " obat yang harus
diminum.")
    else
        output("Kamu lagi ga perlu minum obat kok.")
        return
    endif

while (pasien.jumlahObat > 0) do
    if (pasien.status == butuhPenawar) then
        *valid ← false
        do
            output("Kamu lagi keracunan gara-gara urutan minumannya
salah.. Mau minum penawarnya dulu gak? (y/n): ")
            input(konfirmasi)
            if (konfirmasi == 'n' or konfirmasi == 'N') then
                valid ← true
            else if (konfirmasi == 'y' or konfirmasi == 'Y') then
                output("Oke, minum penawarnya dulu ya!..")
                return
            else
                output("Input tidak valid. Silakan coba lagi.")
            endif
        while (not valid)
    endif

    output("===== DAFTAR OBAT =====")
    output("Urutan minum obat sesuai resep dokter:")
```

```

i traversal [0..pasien.jumlahObatResep-1]
    output(pasien.daftarObatResep[i].nama)
    if (i < pasien.jumlahObatResep - 1) then
        output(" -> ")
    endif
output new line

output("Obat yang harus diminum:")
i traversal [0..pasien.jumlahObat-1]
    output(i+1, ". ", pasien.daftarObat[i].nama)
output new line

output("Obat yang sudah diminum:")
if (stack_is_empty(*perutPasien)) then
    output("Belum ada obat yang diminum.")
else
    i traversal [0..perutPasien.idxTop]
        output(perutPasien.buffer[i].nama)
        if (i < perutPasien.idxTop) then
            output(" -> ")
        endif
    endif
output("Pilih obat untuk diminum: ")
input(pilihanObat)

if (pilihanObat < 1 or pilihanObat > pasien.jumlahObat) then
    output("Obatnya gak diresepkan buat kamu!")
    continue
endif

idxObat ← pilihanObat - 1
stack_push(perutPasien, pasien.daftarObat[idxObat])
output("GLEKGLEKGLEK.. ", pasien.daftarObat[idxObat].nama, " berhasil diminum!")

i traversal [idxObat..pasien.jumlahObat-2]
    pasien.daftarObat[i] ← pasien.daftarObat[i+1]
    pasien.jumlahObat ← pasien.jumlahObat - 1

i traversal [0..perutPasien.idxTop]
    if (strcmp(perutPasien.buffer[i].nama,
    pasien.daftarObatResep[i].nama) ≠ 0) then
        pasien.status ← butuhPenawar
        output("Kamu merasa makin gak enak badan...kayaknya kamu salah minum urutan obat deh..")
        output("Dok! Butuh obat penawar!!")
return

```

```

        endif
    endtraversal

    if (pasien.jumlahObat == 0) then
        if (pasien.status == butuhPenawar) then
            output("Kamu sudah minum semua obat, tapi masih butuh
penawar.")
            return
        else
            pasien.status ← butuhPulang
            output("Kamu sudah minum semua obat yang diberikan dokter.
Silakan pulang ya!")
            break
        endif
    endif
endwhile

```

17. F17 - Minum Penawar

procedure minumPenawar(*input/output*: currUser, *input/output*:
Obat: daftarObat, *output*: Pasien: perutPasien)

{ IS: *Inventory* pasien berisikan daftar obat yang dimiliki oleh
pasien pada saat ini, disusun dalam bentuk List. Serta obat-obat yang
sudah diminum oleh pasien disusun dalam bentuk Stack}

{ FS: Jika Stack konsumsi tidak kosong maka:

- Obat yang terakhir diminum pasien (top of stack) dikeluarkan
- Obat tersebut ditambahkan kembali ke List *inventory* di akhir
- Program menampilkan bahwa obat telah dipindahkan ke *inventory*
kembali

Jika Stack konsumsi kosong:

- Tidak ada perubahan pada konsumsi dan *inventory*
- Program menampilkan bahwa belum ada obat yang dimakan}

KAMUS LOKAL

currUser, userPasien : User
perutPasien : Stack
pasien : Pasien
perutPasien : Stack
urutanBenar, i, n : int
obatTerakhir : Obat

ALGORITMA

```

    if (currUser == NULL) then
        output("Kamu belum login. Silakan login terlebih dahulu dengan
command LOGIN.")
        return
    endif

```

```

if (currUser.role != ROLE_PASIEN) then
    output("Eits...pasien doang yang bisa minum penawar!")
    return
endif

*pasien ← user_pasien.dataPasien
*perutPasien ← &pasien.perutPasien

if (stack_is_empty(*perutPasien)) then
    output("Perut kosong!! Belum ada obat yang dimakan.")
    return
endif

urutanBenar ← false

while (not urutanBenar and not stack_is_empty(*perutPasien)) do
    stack_pop(perutPasien, obatTerakhir)
    pasien.daftarObat[pasien.jumlahObat] ← obatTerakhir
    pasien.jumlahObat ← pasien.jumlahObat + 1

    output("Uwekkk!!! ", obatTerakhir.nama, " keluar dan kembali ke
inventory")

    urutanBenar ← true
    n ← perutPasien.idxTop + 1

    i traversal [0..n-1]
        if (strcmp(pasien.daftarObatResep[i].nama,
perutPasien.buffer[i].nama) ≠ 0) then
            urutanBenar ← false
            break
        endif
    endtraversal

    if (not urutanBenar and not stack_is_empty(*perutPasien)) then
        output("Urutan obat masih belum sesuai resep dokter! Muntahkan
lagi satu obat...")
        endif
    endwhile

    if (urutanBenar) then
        output("Urutan obat terdepan sudah sesuai resep dokter!")
        pasien.status ← butuhMinumObat
    endif

```

18. F18 - Exit

procedure exit_system()

```

{Prosedur untuk keluar dari sistem rumah sakit}
I.S. Program dalam keadaan berjalan, current_user mungkin sedang login
atau tidak
F.S. Jika user sedang login, sistem meminta konfirmasi keluar dan
menutup program jika dikonfirmasi; jika tidak login, sistem meminta
login terlebih dahulu.

```

KAMUS LOKAL
input : string[10]

```

ALGORITMA
if current_user == NULL then
    output ("Tidak ada pengguna yang sedang login. Silakan login
terlebih dahulu untuk keluar.")
else
    output ("Apakah Anda mau keluar dari rumah sakit? (y/n)")
    repeat
        output ("Pilihan Anda: ")
        input (input)

        if (input == "Y" or input == "y") then
            output (current_user.username, " keluar dari Rumah
Sakit,")
            output ("Sampai jumpa!")
            exit(0)
        else if (input == "N" or input == "n") then
            output ("Kembali ke menu utama...")
            return
        else
            output ("Tidak valid. Pilih Y atau N ya!")
    forever
endif

```

19. B06 - Mainin Antrian

- Skip Antrian

```

PROCEDURE skip_antrian (input/output user: User, input/output:
ruangan, input banyakUser: integer)
{ IS:
    - User pasien sedang login
    - Pasien terdaftar dalam antrian dokter
    - Ruangan dokter valid
}
{ FS:
    - Posisi pasien dipindah ke depan antrian (setelah pasien di dalam
ruangan)
    - Posisi antrian lain menyesuaikan
    - Data antrian di-update
}

```

```
}
```

KAMUS LOKAL

```
pasien : Pasien
idxRuang, maxDalamRuang, pos : integer
antrian : AntrianDokter
prev, curr, prevskip, firstAntrian : address
```

ALGORITMA

```
if (currUser == NULL) then
    output("ERROR: Kamu belum login. Silakan login terlebih dahulu
dengan command LOGIN.")
    return
endif

if (user.role != ROLE_PASIEN) then
    output("ERROR: Hanya pasien yang bisa melewati antrian!")
    return
endif

*pasien ← user.dataPasien
if (pasien.posisiAntrian < 0) then
    output("ERROR: Kamu belum terdaftar dalam antrian check-up.")
    return
endif

// Cek validitas ruangan
idxRuang ← pasien.idRuang - 1
if (idxRuang < 0) or (idxRuang >= ruangan.jumlah) then
    output("ERROR: Ruangan tidak ditemukan!")
    return
endif

*antrian ← &ruangan.ruang[idxRuang].Antrian
maxDalamRuang ← ruangan.ruang[idxRuang].kapasitas // Asumsi:
field kapasitas ada
prev ← NULL
curr ← antrian.first
pos ← 1
prevSkip ← NULL // Node sebelum posisi skip (N)
firstAntrian ← NULL // Node ke-(N+1)

// Cari posisi pasien dan node ke-(N+1)
while (curr != NULL) do
    if (pos == maxDalamRuang) then
        prevSkip ← curr
        firstAntrian ← curr.next
    endif
endif
```

```
        if (curr.pasien.dataPasien == pasien) then
            break
        endif
        prev ← curr
        curr ← curr.next
        pos ← pos + 1
    endwhile

    if (curr == NULL) then
        output("ERROR: Kamu tidak berada di antrian.")
        return
    endif

    if (pos <= maxDalamRuangan) then
        output(">> Kamu sedang berada di dalam ruangan, tidak bisa
skip antrian.")
        return
    endif

    if (curr == firstAntrian) then
        output(">> Kamu sudah berada di posisi pertama antrian.")
        return
    endif

    // Hapus dari posisi lama
    if (prev != NULL) then
        prev.next ← curr.next
        if (curr == antrian.last) then
            antrian.last ← prev
        endif
    endif

    // Masukkan ke posisi pertama antrian (setelah pasien di ruangan)
    if (prevSkip != NULL) then
        curr.next ← prevSkip.next
        prevSkip.next ← curr
        if (curr.next == NULL) then
            antrian.last ← curr
        endif
    endif

    output(">> Kamu berhasil pindah ke posisi pertama antrian!")
```

I. LAMPIRAN SOURCE CODE

Gambar 21 Source code struktur data user.h

```
header > C user.h > 68_unnamed_struct_04a1_2 > username
1 #ifndef USER_H
2 #define USER_H
3
4 #include <stdio.h>
5 #include <string.h>
6 #include "boolean.h"
7 #include "pasien.h"
8 // #include "dokter.h"
9
10 #define MAX_USERS 100
11 #define USERNAME_LEN 50
12 #define PASSWORD_LEN 50
13
14 typedef struct Dokter Dokter;
15
16 // Enum untuk role pengguna
17 typedef enum {
18     ROLE_MANAGER,
19     ROLE_DOKTER,
20     ROLE_PASIEN,
21     ROLE_NONE
22 } UserRole;
23
24 // Struktur data user
25 typedef struct {
26     char username[USERNAME_LEN];
27     char password[PASSWORD_LEN];
28     UserRole role;
29     int idUser; // ID unik untuk setiap user
30
31     union {
32         Pasien *dataPasien; // Jika role == ROLE_PASIEN
33         Dokter *dataDokter; // Jika role == ROLE_DOKTER
34         // Manager *manager_data; // Jika role == ROLE_MANAGER
35     };
36 } User;
37
38 // Variabel global yang digunakan di seluruh program
39 extern int user_count; // Jumlah user yang terdaftar
40 extern User *currUser; // Pointer ke user yang sedang login
41
42 // Deklarasi fungsi utility
43 // Di user.h
44 // Deklarasi saja (tanpa implementasi)
45 const char* role_to_string(UserRole role);
46
47 boolean is_alpha_string(const char* str); // Deklarasi saja!
48 #endif
```

Gambar 22 Source code header ADT list_user.h

```
#ifndef List_User_H
#define List_User_H

#include "Boolean.h"
#include "../../header/user.h"

#define MAX_CAPACITY 100
#define IDX_UNDEF -999 /* Tidak terdefinisi */

typedef int IdxType;

typedef struct {
    User data[MAX_CAPACITY];
    IdxType length;
} ListUser;

/* Indeks yang digunakan [0..MAX_CAPACITY-1] */

/* ***** KONSTRUKTOR ARRAY *****/
/* Konstruktor : create tabel kosong */
void MakeEmpty(ListUser *L);
/* I.S. sembarang */
/* F.S. Terbentuk list L kosong dengan kapasitas MAX_CAPACITY */

/* ***** SELEKTOR *****/
/* *** Banyaknya elemen *** */
int NbElmt(ListUser L);
/* Mengirimkan banyaknya elemen efektif list */
/* Mengirimkan nol jika list kosong */

/* *** Daya tampung container *** */
int MaxNbEl(ListUser L);
/* Mengirimkan maksimum elemen yang dapat ditampung oleh list */

/* *** Selektor INDEKS *** */
IdxType GetFirstIdx(ListUser L);
/* Prekondisi : List L tidak kosong */
/* Mengirimkan indeks elemen pertama */

IdxType GetLastIdx(ListUser L);
/* Prekondisi : List L tidak kosong */
/* Mengirimkan indeks elemen terakhir */

/* *** Menghasilkan sebuah elemen *** */
User GetElmt(ListUser L, IdxType i);
/* Prekondisi : List tidak kosong, i antara FirstIdx(L)..LastIdx(L) */
/* Mengirimkan elemen list yang ke-i */
```

```

/* *** Selektor SET : Mengubah nilai list dan elemen list *** */
void SetTab(ListUser Lin, ListUser *Lout);
/* I.S. Lin terdefinisi, sembarang */
/* F.S. Lout berisi salinan Lin */
/* Assignment Lout = Lin */

void SetEl(ListUser *L, IdxType i, User v);
/* I.S. L terdefinisi, sembarang */
/* F.S. Elemen L yang ke-i bernilai v */
/* Mengeset nilai elemen list yang ke-i sehingga bernilai v */

void SetLength(ListUser *L, IdxType N);
/* I.S. L terdefinisi, sembarang */
/* F.S. Nilai indeks efektif L bernilai N */
/* Mengeset nilai i lai N */

/* ***** Test ✨ Generate Copilot summary
boolean IsIdxValid(ListUser L, IdxType i);
/* Prekondisi : i sembarang */
/* Mengirimkan true jika i adalah indeks yang valid utk ukuran list */
/* yaitu antara indeks yang terdefinisi utk container */

boolean IsIdxEff(ListUser L, IdxType i);
/* Prekondisi : i sembarang*/
/* Mengirimkan true jika i adalah indeks yang terdefinisi utk list */
/* yaitu antara GetFirstIdx(L)..GetLastIdx(L) */

/* ***** TEST KOSONG/PENUH *****/
/* *** Test tabel kosong *** */
boolean IsEmpty(ListUser L);
/* Mengirimkan true jika list L kosong, mengirimkan false jika tidak */

/* *** Test tabel penuh *** */
boolean IsFull(ListUser L);
/* Mengirimkan true jika tabel T penuh, mengirimkan false jika tidak */

#endif

```

Gambar 23 Source code implementasi ADT list_user.h

```
#include "../header/List_User.h"

void MakeEmpty(ListUser *L) {
    L->length = 0;
}

int NbElmt(ListUser L) {
    return L.length;
}

int MaxNbEl(ListUser L) {
    return MAX_CAPACITY;
}

IdxType GetFirstIdx(ListUser L) {
    return 0;
}

IdxType GetLastIdx(ListUser L) {
    return L.length - 1;
}

User GetElmt(ListUser L, IdxType i) {
    return L.data[i];
}

void SetTab(ListUser Lin, ListUser *Lout) {
    Lout->length = Lin.length;
    for (int i = 0; i < Lin.length; i++) {
        Lout->data[i] = Lin.data[i];
    }
}

void SetEl(ListUser *L, IdxType i, User v) {
    L->data[i] = v;
    if (L->length < i + 1) {
        L->length = i + 1;
    }
}

void SetLength(ListUser *L, IdxType N) {
    L->length = N;
}

boolean IsIdxValid(ListUser L, IdxType i) {
    return i >= GetFirstIdx(L) && i < MAX_CAPACITY;
}

boolean IsIdxEff(ListUser L, IdxType i) {
    return i >= GetFirstIdx(L) && i <= GetLastIdx(L);
}

boolean IsEmpty(ListUser L) {
    return L.length == 0;
}

boolean IsFull(ListUser L) {
    return L.length == MAX_CAPACITY;
}
```

Gambar 24 Source code header ADT set.h

```
#ifndef SET_H
#define SET_H

#include "boolean.h"

#define Nil 0
#define MaxEl 100
#define USERNAME_LEN 50 // Panjang maksimum username

typedef struct {
    char Elements[MaxEl][USERNAME_LEN]; // Array untuk menyimpan username
    int Count; // Jumlah elemen dalam Set
} Set;

extern Set usernames; // Deklarasi variabel global untuk menyimpan daftar username

/* Definisi Set S kosong : S.Count = Nil */

/* *** Konstruktor/Kreator ***
void CreateEmpty(Set *S);
/* I.S. Sembarang */
/* F.S. Membuat sebuah Set S kosong berkapasitas MaxEl */
/* Ciri Set kosong : count bernilai Nil */

/* ***** Predikat Untuk test keadaan KOLEKSI *****/
boolean Set_IsEmpty(Set S);
/* Mengirim true jika Set S kosong */
/* Ciri Set kosong : count bernilai Nil */

boolean Set_IsFull(Set S);
/* Mengirim true jika Set S penuh */
/* Ciri Set penuh : count bernilai MaxEl */

/* ***** Operator Dasar Set *****/
void Insert(Set *S, const char *Elmt);
/* Menambahkan Elmt sebagai elemen Set S. */
/* I.S. S mungkin kosong, S tidak penuh
   |   S mungkin sudah beranggotakan Elmt */
/* F.S. Elmt menjadi anggota dari S. Jika Elmt sudah merupakan anggota, operasi tidak dilakukan */

void Delete(Set *S, const char *Elmt);
/* Menghapus Elmt dari Set S. */
/* I.S. S tidak kosong
   |   Elmt mungkin anggota / bukan anggota dari S */
/* F.S. Elmt bukan anggota dari S */

boolean IsMember(Set S, const char *Elmt);
/* Mengembalikan true jika Elmt adalah anggota dari S */

boolean is_username_unique(const Set *usernames, const char *username);

#endif
```

Gambar 25 Source code implementasi ADT set.h

```
#include "../header/set.h"
#include <stdio.h>
#include <string.h> // Untuk operasi string

void CreateEmpty(Set *S) {
    S->Count = Nil;
}

boolean Set_IsEmpty(Set S) {
    return S.Count == Nil;
}

boolean Set_IsFull(Set S) {
    return S.Count == MaxEl;
}

void Insert(Set *S, const char *Elmt) {
    if (Set_IsFull(*S)) return; // Tidak bisa menambahkan jika penuh
    if (IsMember(*S, Elmt)) return; // Tidak menambahkan elemen duplikat

    strcpy(S->Elements[S->Count], Elmt); // Salin elemen ke dalam Set
    S->Count++;
}

void Delete(Set *S, const char *Elmt) {
    if (Set_IsEmpty(*S)) return; // Tidak bisa menghapus jika kosong

    int idx = -1;
    for (int i = 0; i < S->Count; i++) {
        if (strcmp(S->Elements[i], Elmt) == 0) { // Bandingkan string
            idx = i;
            break;
        }
    }

    if (idx == -1) return; // Elemen tidak ditemukan

    for (int i = idx; i < S->Count - 1; i++) {
        strcpy(S->Elements[i], S->Elements[i + 1]); // Geser elemen
    }
    S->Count--;
}

boolean IsMember(Set S, const char *Elmt) {
    for (int i = 0; i < S.Count; i++) {
        if (strcmp(S.Elements[i], Elmt) == 0) { // Bandingkan string
            return true;
        }
    }
    return false;
}
```

Gambar 26 Source code F01: login.c

```
#include "../header/login.h"
#include <stdio.h>
#include <string.h>

void login_system(ListUser *users) {
    // KAMUS LOKAL
    boolean found_user = false, valid_password = false; // Apakah username ditemukan, Apakah password valid
    char username[USERNAME_LEN]; // Input username
    char password[PASSWORD_LEN]; // Input password
    char role[20]; // Role pengguna
    int i; // Indeks iterasi

    // Cek apakah sudah ada user yang login
    if (current_user != NULL) {
        printf("Anda sudah login sebagai %s. Silakan logout terlebih dahulu untuk login dengan akun lain.\n",
               role_to_string(current_user->role), current_user->username);
        return;
    }

    // Input username
    printf("Masukan username: ");
    if (scanf("%49s", username) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    // Input password
    printf("Masukan password: ");
    if (scanf("%49s", password) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    // Iterasi untuk mencari username
    for (i = GetFirstIdx(*users); i <= GetLastIdx(*users); i++) {
        User user = GetElmt(*users, i);

        if (strcmp(user.username, username) == 0) {
            found_user = true; // Username ditemukan

            if (strcmp(user.password, password) == 0) {
                valid_password = true; // Password cocok
                current_user = &users->data[i];

                // Tampilkan pesan selamat datang
                printf("Selamat pagi %s!", role_to_string(current_user->role),
                       current_user->username);

                if (current_user->role == ROLE_PASIEN) {
                    printf(" Ada keluhan apa ?\n");
                } else {
                    printf("\n");
                }
                return;
            }
            if (!valid_password) { // Password tidak cocok
                printf("Password salah untuk pengguna yang bernama %s!\n", username);
                return;
            }
        }
    }

    if (!found_user) {
        printf("Tidak ada Manager, Dokter, atau pun Pasien yang bernama %s!\n", username);
    }
}
```

Gambar 27 Source code F02: register.c

```
#include "../header/register.h"
#include "../header/user.h"
#include <stdio.h>
#include <string.h>

void register_pasien(ListUser *users, Set *usernames) {
    if (current_user != NULL) {
        printf("Anda sudah login. Silakan logout terlebih dahulu.\n");
        return;
    }

    char username[USERNAME_LEN];
    char password[PASSWORD_LEN];

    printf("Username: ");
    if (scanf("%49s", username) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    if (!is_alpha_string(username)) {
        printf("Username hanya boleh berisi huruf!\n");
        return;
    }

    if (!is_username_unique(usernames, username)) {
        printf("Registrasi gagal! Username %s sudah digunakan!\n", username);
        return;
    }

    printf("Password: ");
    if (scanf("%49s", password) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    if (IsFull(*users)) {
        printf("Kapasitas user penuh!\n");
        return;
    }

    User new_user;
    strcpy(new_user.username, username);
    strcpy(new_user.password, password);
    new_user.role = ROLE_PASIEN;

    SetEl(users, NbElmt(*users), new_user);
    SetLength(users, NbElmt(*users) + 1);
    Insert(usernames, username);

    printf("Pasien %s berhasil ditambahkan!\n", username);
}
```

Gambar 28 Source code F03: logout.c

```
#include "../header/logout.h"
#include <stdio.h>

void logout_system() {
    boolean is_logged_in = (current_user != NULL);
    if (!is_logged_in) {
        printf("Logout gagal!\n");
        printf("Anda belum login, silahkan login terlebih dahulu sebelum melakukan logout\n");
        return;
    }
    printf("Sampai jumpa %s!\n", current_user->username);
    current_user = NULL;
}
```

Gambar 29 Source code F04: lupa_password.c

```
void lupa_password_system(listUser *users) {
    char username[USERNAME_LEN];
    char kode_unik_input[100];
    char kode_unik_asli[100];
    char password_baru[PASSWORD_LEN];

    printf("Username: ");
    if (scanf("%49s", username) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    User* user = NULL;
    for (int i = GetFirstIdx(*users); i <= GetLastIdx(*users); i++) {
        if (strcmp(GetIdx(*users, i).username, username) == 0) {
            user = &users->data[i];
            break;
        }
    }

    if (user == NULL) {
        printf("Username tidak terdaftar!\n");
        return;
    }

    printf("Kode Unik: ");
    if (scanf("%99s", kode_unik_input) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    generate_kode_unik(user->username, kode_unik_asli);

    if (strcmp(kode_unik_input, kode_unik_asli) != 0) {
        printf("Kode unik salah!\n");
        return;
    }

    printf("Halo %s, silakan daftarkan ulang password anda!\n",
           role_to_string(user->role), user->username);
    printf("Password Baru: ");
    if (scanf("%49s", password_baru) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    if (strcmp(password_baru, user->password) == 0) {
        printf("Password baru tidak boleh sama dengan password lama!\n");
        return;
    }

    strcpy(user->password, password_baru);
    printf("Password berhasil diubah!\n");
}
```

Gambar 30 Source code F05: help.c

```
#include "../header/help.h"
#include <stdio.h>
|
void help_system() {
    printf("\n===== HELP =====\n\n");
    if (current_user == NULL) {
        printf("Kamu belum login sebagai role apapun. Silahkan login terlebih dahulu.\n\n");
        printf("LOGIN: Masuk ke dalam akun yang sudah terdaftar\n");
        printf("REGISTER: Membuat akun baru\n");
    } else {
        printf("Halo %s . ", role_to_string(current_user->role), current_user->username);
        if (current_user->role == ROLE_DOKTER) {
            printf("Kamu memanggil command HELP. Kamu pasti sedang kebingungan.\n");
            printf("Berikut adalah hal-hal yang dapat kamu lakukan sekarang:\n\n");
            printf("LOGOUT: Keluar dari akun yang sedang digunakan\n");
            printf("DIAGNOSIS: Melakukan diagnosis penyakit pasien berdasarkan kondisi tubuh pasien\n");
        } else if (current_user->role == ROLE_PASTIEN) {
            printf("Kamu memanggil command HELP. Kamu pasti sedang kebingungan.\n");
            printf("Berikut adalah hal-hal yang dapat kamu lakukan sekarang:\n\n");
            printf("LOGOUT: Keluar dari akun yang sedang digunakan\n");
            printf("DAFTAR_CHECKUP: Mendaftarkan diri untuk pemeriksaan dokter\n");
        } else if (current_user->role == ROLE_MANAGER) {
            printf("Kenapa kamu memanggil command HELP? Kamu manager, ");
            printf("tapi yasudahlah kamu pasti sedang kebingungan.\n");
            printf("Berikut adalah hal-hal yang dapat kamu lakukan sekarang:\n\n");
            printf("LOGOUT: Keluar dari akun yang sedang digunakan\n");
            printf("TAMBAH_DOKTER: Mendaftarkan dokter baru ke sistem\n");
        }
    }
    printf("\nFootnote: \n");
    printf("Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar\n");
    printf("Jangan lupa untuk memasukkan input yang valid\n");
}
```

Gambar 31 Source code F06: lihat_denah.c

```
#include <stdio.h>
#include "../header/lihat_denah.h"
|
void lihat_denah (ListRuang ruangan) {
    printf("\n");
    printf("+---+---+---+---+---+---+---+\n");
    for (int i = 0; i < ruangan.jumlah; i++) {
        printf("| %d ", ruangan.ruang[i].nomor);
    }
    printf("|\n");
    printf("+---+---+---+---+---+---+---+\n");
}
|
void lihat_ruangan (ListRuang ruangan, int num){
    while (current_user->role != ROLE_MANAGER && current_user->role != ROLE_DOKTER
          && current_user->role != ROLE_PASTIEN) {
        printf ("Role tidak ditemukan. Akses ditolak.\n");
        return;
    }

    if (num <= 0 || num > ruangan.jumlah) {
        printf("Nomor ruangan tidak valid.\n");
        return;
    }

    Ruangan *r = &ruangan.ruang[num-1];

    printf("\n--- Detail Ruangan %d ---\n", r->nomor);
    printf("Kapasitas : %d\n", r->kapasitas);
    printf("Dokter : %s\n", r->dokter);
    printf("Pasien di dalam ruangan:\n");

    if (r->pasien.Count == 0) {
        printf("Tidak ada pasien di dalam ruangan saat ini.\n");
    } else {
        for (int i = 0; i < r->pasien.Count; i++) {
            printf(" %d. %s\n", i + 1, r->pasien.Elements[i]);
        }
    }
    printf("-----\n\n");
}
```

Gambar 32 Source code F10: f10.c

```
#include <stdio.h>
#include "../header/f10.h"

void tambah_dokter (ListUser *users, Set *usernames){
    if (current_user == NULL) {
        printf("\nTidak ada pengguna yang sedang login. Silakan login terlebih dahulu.\n");
        return;
    }

    if (current_user->role != ROLE_MANAGER){
        printf ("Akses ditolak. Anda bukan Manager.\n");
        return;
    }

    char username[USERNAME_LEN];
    char password[PASSWORD_LEN];

    printf("Username: ");
    if (scanf("%49s", username) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    if (!is_alpha_string(username)) {
        printf("Username hanya boleh berisi huruf!\n");
        return;
    }

    if (!is_username_unique(usernames, username)) {
        printf("Sudah ada dokter bernama %s!\n", username);
        return;
    }

    printf("Password: ");
    if (scanf("%49s", password) != 1) {
        printf("Input tidak valid!\n");
        while (getchar() != '\n');
        return;
    }

    if (IsFull(*users)) {
        printf("Kapasitas user penuh!\n");
        return;
    }

    User new_user;
    strcpy(new_user.username, username);
    strcpy(new_user.password, password);
    new_user.role = ROLE_DOKTER;

    SetEl(users, NbElmt(*users), new_user);
    SetLength(users, NbElmt(*users) + 1);
    Insert(usernames, username); // Tambahkan username ke dalam Set

    printf("Dokter %s berhasil ditambahkan!\n", username);
}
```

Gambar 33 Source code F18: exit.c

```
#include "../header/exit.h" // Untuk akses exit_system
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void exit_system(ListUser *users) {
    // Cek apakah ada pengguna yang sedang login
    if (current_user == NULL) {
        printf("\nTidak ada pengguna yang sedang login. Silakan login terlebih dahulu untuk keluar.\n");
        return;
    }

    char input[10];

    printf("\nApakah Anda mau keluar dari rumah sakit? (y/n)\n");

    while (1) {
        printf("Pilihan Anda: ");
        scanf("%s", input);

        if (strcmp(input, "Y") == 0 || strcmp(input, "y") == 0) {
            printf("\n%s keluar dari Rumah Sakit,\nSampai jumpa!\n", current_user->username);
            exit(0);
        } else if (strcmp(input, "N") == 0 || strcmp(input, "n") == 0) {
            printf("\nKembali ke menu utama...\n");
            return;
        } else {
            printf("Tidak valid. Pilih Y atau N ya!\n");
        }
    }
}
```

Gambar 34 Source code Diagnosis

```

#include <stdio.h>
#include "../header/Diagnosis.h"
#include "../header/Dokter.h"
#include "../DTR/header/Ruang.h"
#include "../header/Pasien.h"

extern ListRuangan ruangan;

static char *identifikasi_penyakit(Pasien *pasien) {
    static char namaPenyakit[MAX PENYAKIT];
    for (int i = 0; i < jumlahPenyakit; i++) {
        Penyakit p = ketPenyakit[i];
        if (pasien->suhu > p.suhuMin && pasien->suhu < p.suhuMax &
            pasien->tekananDarah[0] > p.bpdMin && pasien->tekananDarah[0] < p.bpdMax &&
            pasien->tekananDarah[1] > p.bpdMin && pasien->tekananDarah[1] < p.bpdMax &&
            pasien->detakJantung > p.bpmMin && pasien->detakJantung < p.bpmMax &&
            pasien->saturasiOksigen > p.satMin && pasien->saturasiOksigen < p.satMax &&
            pasien->kadarGulaDarah > p.bgmMin && pasien->kadarGulaDarah < p.bgmMax &&
            pasien->beratBadan > p.bbMin && pasien->beratBadan < p.bbMax &&
            pasien->ttinggiBadan > p.tbMin && pasien->ttinggiBadan < p.tbMax &&
            pasien->kadarKolesterol > p.fatMin && pasien->kadarKolesterol < p.fatMax &&
            pasien->trombosit > p.tromin && pasien->trombosit < p.tromax) {
            strcpy(pasien->penyakit, p.nama);
            strcpy(namaPenyakit[MAX PENYAKIT-1], "\0");
            return namaPenyakit;
        }
    }
    return NULL;
}

void diagnosis_pasien (User *userDokter) {
    if (userDokter == NULL) {
        printf("ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (pasien->status == butuhDiberiObat) {

        if (pasien->status != butuhDiagnosa) {
            printf("[dr. %s] Pasien @%s tidak perlu didiagnosis lagi.\n", userDokter->username, userPasien->username);
            return;
        }

        char *penyakit = identifikasi_penyakit(pasien);
        printf("\n+-----+|-----+|\n");
        printf("          |DIAGNOSA| |\n");
        printf("          +-----+|\n");
        if (penyakit != NULL) {
            pasien->status = butuhDiberiObat;
            printf("[dr. %s] Pasien @%s terdiagnosa mengidap penyakit: %s\n", userDokter->username, userPasien->username, penyakit);
            printf("[dr. %s] Jangan lupa untuk diobatin ya!\n", userDokter->username);
            printf("HELP: Untuk mengobati pasien @%s, ketik NGOBATIN!\n", userPasien->username);
        } else {
            pasien->status = butuhPulang;
            printf("[dr. %s] Pasien @%s sehat banget! Dijamin kuat salto keliling kota!\n", userDokter->username, userPasien->username);
        }
    }
}

if (userDokter->role != ROLE_DOKTER) {
    printf("ERROR: Hanya dokter yang bisa diagnosis!\n");
    return;
}

Dokter *dokter = userDokter->dataDokter;
int idxRuang = dokter->nomorRuang - 1;
for (int i = 0; i < ruangan.jumlah; i++) {
    if (ruangan.ruang[i].dokter != NULL && ruangan.ruang[i].dokter->id == dokter->id) {
        idxRuang = i;
        break;
    }
}

if (idxRuang < 0 || idxRuang > ruangan.jumlah) {
    printf("[dr. %s] Kamu belum memiliki ruangan.\n", userDokter->username);
    return;
}

Ruang *r = &ruangan.ruang[idxRuang];
address current = r->Atrial.first;
if (current == NULL) {
    printf("[dr. %s] Kamu lagi nggak ada pasien. Asik, free time!\n", userDokter->username);
}

User *userPasien = current->pasien;
Pasien *pasien = userPasien->dataPasien;

if (pasien->status == butuhDiberiObat) {
    printf("[dr. %s] Pasien @%s udah didiagnosa, tinggal dikasih obat aja.\n", userDokter->username, userPasien->username);
    return;
}
}

```

Gambar 35 Source code Ngobatin

```
#include "../header/Ngobatin.h"
#include <stdio.h>
#include <string.h>
#include "../AFD/header/Ruang.h"
#include "../header/User.h"
#include "../header/Dokter.h"

PenyakitObatEntry* cari_obat(const char *namaPenyakit) {
    for (int i = 0; i < jumlahPenyakit; i++) {
        if (strcmp(penyakitObatMap[i].namaPenyakit, namaPenyakit) == 0) {
            return &penyakitObatMap[i];
        }
    }
    return NULL; // Tidak ditemukan
}

void ngobatin (User *currUser, User *users, int banyakUser, ListRuang *ruangan) {
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_DOKTER) {
        printf("\nERROR: Hanya dokter yang punya kemampuan mengobati pasien!\n");
        return;
    }

    // Cari data dokter
    Dokter *dokter = currUser->dataDokter;
    if (dokter == NULL) {
        printf("\nERROR: Data dokter tidak valid!\n");
        return;
    }

    if (pasien->status == butuhMinumObat) {
        printf("[dr. %s] Pasien @%s sudah diberi obat, tidak perlu diobati lagi.\n", currUser->username, userPasien->username);
        return;
    }

    PenyakitObatEntry *entry = cari_obat(pasien->penyakit);
    if (entry == NULL) {
        printf("[dr. %s] Tidak ada obat untuk penyakit %s!\n", currUser->username, pasien->penyakit);
        return;
    }

    printf("[dr. %s] Resep untuk @%s (Penyakit: %s):\n", currUser->username, userPasien->username, pasien->penyakit);
    for (int i = 0; i < entry->jumlahObat; i++) {
        printf("      %d. %s\n", i+1, entry->obat[i].nama);
    }

    // Simpan daftar obat ke pasien
    pasien->jumlahObat = entry->jumlahObat;
    pasien->jumlahObatResep = entry->jumlahObat;

    for (int i = 0; i < entry->jumlahObat; i++) {
        pasien->daftarObat[i] = entry->obat[i];
        pasien->daftarObatResep[i] = entry->obat[i];
    }
    pasien->status = butuhMinumObat;
}

// Cari ruangan dokter
Ruang *ruanganDokter = NULL;
for (int i = 0; i < ruangan->jumlah; i++) {
    if (ruangan->ruang[i].dokter != NULL && ruangan->ruang[i].dokter->id == dokter->id) {
        ruanganDokter = &ruangan->ruang[i];
        break;
    }
}

printf("-----\n");
printf("          NOGABATIN | \n");
printf("-----\n");

if (ruanganDokter == NULL) {
    printf("[dr. %s] Kamu belum memiliki ruangan!\n", currUser->username);
    return;
}

Pasien *pasien = current->pasien->dataPasien;
User *userPasien = current->pasien->pasien;

if (pasien->status == butuhDiagnosa) {
    printf("[dr. %s] Pasien @%s belum didiagnosis. Diagnosis dulu sebelum mengobati!\n", currUser->username, userPasien->username);
    return;
}
if (pasien->status == butuhPulang) {
    printf("[dr. %s] Pasien @%s sudah sembuh, tidak perlu diobati lagi.\n", currUser->username, userPasien->username);
    return;
}
```

Snipping Tool

Gambar 36 Source code Daftar Check Up

```
#include "../header/Daftar_Check_Up.h"
#include <stdio.h>
#include <string.h>
#include "../header/User.h"
#include "../header/Dokter.h"
#include "../header/Ruang.h"

// Validasi input angka positif
boolean validasi_float (float nilai, const char* namavar) {
    if (nilai <= 0) {
        printf("\nERROR: %s harus positif!\n", namavar);
        return false;
    }
    return true;
}

boolean validasi_integer (int nilai, const char* namavar) {
    if (nilai <= 0) {
        printf("\nERROR: %s harus positif!\n", namavar);
        return false;
    }
    return true;
}

// Tampilkan daftar dokter yang tersedia
void display_dokter (User *users, int banyakUser, ListRuang *ruangan) {
    printf("\nBerikut adalah daftar dokter yang tersedia:\n");
    int banyakDokter = 0;

    for (int i = 0; i < banyakUser; i++) {
        if (users[i].role != ROLE_DOKTER) {
            continue;
        }

        if (users[i].dataDokter == NULL) {
            printf("\nERROR: Data dokter tidak valid untuk dr. %s!\n", users[i].username);
            continue; // skip invalid entries
        }
    }
}

// Cari ruangan dokter
Ruang *ruanganDokter = NULL;
for (int i = 0; i < ruangan->jumlah; i++) {
    if (ruangan->ruang[i].dokter != NULL && ruangan->ruang[i].dokter->id == dokter->id) {
        ruanganDokter = &ruangan->ruang[i];
        break;
    }
}

printf("%d. dr. %s (%s) - Total Ruangan + Antrian: %d\n",
    banyakDokter,
    users[i].username,
    dokter->nomorRuang,
    ruanganDokter->Antrian.jumlah);

void daftar_check_up(User *currUser, User *users, int banyakUser, ListRuang *ruangan) {
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }
```

```

72     if (currUser->role != ROLE_PASIEN) {
73         printf("\nERROR: Hanya pasien yang bisa daftar check-up.\n");
74         return;
75     }
76
77     if (currUser->dataPasien == NULL) {
78         printf("\nERROR: Data pasien tidak valid!\n");
79         return;
80     }
81
82     Pasien *pasien = currUser->dataPasien;
83
84     // Cek apakah pasien sudah dalam antrian
85     if (pasien->posisiAntrian >= 0) {
86         printf("[@%s] Kamu sudah terdaftar di antrian!\n", currUser->username);
87         printf(" Silakan selesaikan check-up yang sudah terdaftar terlebih dahulu.\n");
88         return;
89     }
90
91     //cek apakah pasien sudah berada di ruangan/antrian (berdasarkan queue, bukan array manual)
92     for (int i = 0; i < ruangan->jumlah; i++) {
93         Ruangan *r = &ruangan->ruangan[i];
94         address curr = r->antrian.first;
95         while (curr != NULL) {
96             if (curr->pasien->dataPasien == pasien) {
97                 printf("[@%s] Kamu sudah berada di ruangan %d (atau antrian ruangan %d)\n",
98                     currUser->username, r->nomor, r->nomor);
99                 return;
100            }
101            curr = curr->next;
102        }
103    }
104
105    // Hitung jumlah dokter yang tersedia
106    int banyakDokter = 0;
107    for (int i = 0; i < banyakUser; i++) {
108        if (users[i].role == ROLE_DOKTER) {
109            banyakDokter++;
110        }
111    }
112
113    printf("\n+-----\n| DAFTAR CHECK UP |\n+-----\n");
114
115    printf(">> Input data medis:\n");
116    do {
117        printf("Suhu tubuh (Celcius): ");
118        scanf("%f", &pasien->suhu);
119    } while (!validasi_float(pasien->suhu, "Suhu"));
120    do {
121        printf("Tekanan darah (sistol/diastol, contoh: 120 80): ");
122        scanf("%d %d", &pasien->tekananDarah[0], &pasien->tekananDarah[1]);
123    } while (pasien->tekananDarah[0] <= 0 || pasien->tekananDarah[1] <= 0);
124    do {
125        printf("Detak jantung (bpm): ");
126        scanf("%d", &pasien->detakJantung);
127    } while (!validasi_integer(pasien->detakJantung, "Detak jantung"));
128    do {
129        printf("Saturasi oksigen (persentase): ");
130        scanf("%f", &pasien->saturasiOksigen);
131    } while (!validasi_float(pasien->saturasiOksigen, "Saturasi oksigen"));
132    do {
133        printf("Kadar gula darah (mg/dL): ");
134        scanf("%d", &pasien->kadarGulaDarah);
135    } while (!validasi_integer(pasien->kadarGulaDarah, "Kadar gula darah"));

136
137
138    do {
139        printf("Berat badan (kg): ");
140        scanf("%f", &pasien->beratBadan);
141    } while (!validasi_float(pasien->beratBadan, "Berat badan"));
142    do {
143        printf("Tinggi badan (cm): ");
144        scanf("%d", &pasien->tinggiBadan);
145    } while (!validasi_integer(pasien->tinggiBadan, "Tinggi badan"));
146    do {
147        printf("Kadar kolesterol (mg/dL): ");
148        scanf("%d", &pasien->kadarKolesterol);
149    } while (!validasi_integer(pasien->kadarKolesterol, "Kadar kolesterol"));
150    do {
151        printf("Trombosit (ribu/µL): ");
152        scanf("%d", &pasien->trombosit);
153    } while (!validasi_integer(pasien->trombosit, "Trombosit"));

154
155    display_dokter(users, banyakUser, ruangan);

156    int pilihan;
157    for (;;) {
158        printf("\nPilih dokter (1-%d): ", banyakDokter);
159        scanf("%d", &pilihan);
160        if (pilihan >= 1 && pilihan <= banyakDokter) {
161            break;
162        }
163        printf("ERROR: Pilihan harus berada antara 1 sampai %d!\n", banyakDokter);
164    }
165
166    // Cari dokter yang dipilih
167    int idxDokter = 0;
168    Dokter *dokterPilihan = NULL;
169    User *userDokter = NULL;
170
171
172    printf("\n>> Pendaftaran berhasil! <<\n");
173    printf("[@%s] Kamu terdaftar pada antrian dr. %s di ruangan %d.\n",
174           currUser->username,
175           userDokter->username,
176           dokterPilihan->nomorRuangan);
177    printf("[@%s] Posisi antrian Kamu: %d\n",
178           currUser->username, result->posisiAntrian);
179
180    } else {
181        printf("ERROR: Gagal mendaftar!\n");
182    }
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219

```

Gambar 37 Source code Antrian Saya!

```

1 #include "../header/Antrian_Saya.h"
2 #include <stdio.h>
3 #include <string.h>
4
5 void cek_antrian_saya (User *user, User *users, ListRuangan *ruangan, int banyakUser) {
6     if (currUser == NULL) {
7         printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
8         return;
9     }
10
11     if (user->role != ROLE_PASIEN) {
12         printf("\nERROR: Hanya pasien yang bisa melihat antrian!\n");
13         return;
14     }
15
16     Pasien *pasien = user->dataPasien;
17
18     // Cek apakah pasien ada di antrian/ruangan manapun (berdasarkan queue)
19     int found = 0;
20     int posisi = 0;
21     int idxRuang = -1;
22     for (int i = 0; i < ruangan->jumlah; i++) {
23         antrian *ruangan = ruangan->ruang[i].Antrian.first;
24         posisi = 0;
25         while (curr != NULL) {
26             if (curr->pasien->dataPasien == pasien) {
27                 found = 1;
28                 idxRuang = i;
29                 break;
30             }
31             curr = curr->next;
32             posisi++;
33         }
34         if (found) {
35             break;
36         }
37     }
38
39     if (!found) {
40         if (pasien->posisiAntrian < 0) {
41             printf("[@%s] Kamu nih belum terdaftar dalam antrian check-up (berkata dengan nada lemah lembut gemas)\nHEP: Silakan daftar terlebih dahulu dengan command DAFTAR CHECKUP.\n");
42         }
43         printf("\n");
44         return;
45     }
46
47     // Jika posisi < MAX_PASIEN_RUANGAN, berarti sudah di ruangan
48     if (posisi < MAX_PASIEN_RUANGAN) {
49         printf("[@%s] Kamu lagi di ruangan dokter, loh. Dokternya ga lagi tidur kan?\n", user->username);
50         return;
51     }
52
53     // Jika posisi >= MAX_PASIEN_RUANGAN, tampilkan status antrian
54     // Cari dokter dari id ruangan
55     if (idxRuang < 0 || idxRuang > ruangan->jumlah) {
56         printf("\nERROR: Ruangan tidak ditemukan!\n");
57         return;
58     }
59     if (ruangan->ruang[idxRuang].Antrian.first == NULL) {
60         printf("\nERROR: Antrian ruangan %d kosong!\n", ruangan->ruang[idxRuang].nomor);
61         return;
62     }
63     // Cari dokter yang menangani pasien ini
64     Ruangan *ruanganDokter = Ruangan->ruang[idxRuang];
65     Dokter *dokterPasien = ruanganDokter->dokter;
66
67     if (dokterPasien == NULL) {
68         printf("\nERROR: Dokter pasien %s tidak ditemukan!\n", user->username);
69         return;
70     }
71     int totalAntrian = queue_size(&ruangan->ruang[idxRuang].Antrian);
72
73     printf("\n+-----+ LIHAT ANTRIAN +-----+\n");
74     printf("|\n");
75     printf("+-+-----+-----+-----+\n");
76
77     printf("[@%s] Status antrian Anda:\n", user->username);
78     printf("    Dokter: dr. %s\n", dokterPasien->username);
79     printf("    Ruangan: %d\n", dokterPasien->nomorRuangan);
80     printf("    Posisi antrian: %d dari %d\n", posisi - MAX_PASIEN_RUANGAN + 1,
81           totalAntrian - MAX_PASIEN_RUANGAN);
82 }
```

Gambar 38 Source code Struktur Data Pasien

```

#ifndef PASIEN_H
#define PASIEN_H

#include "boolean.h"
#include "obat.h"
#include "../ADT/header/Stack.h"

#define MAX_PENYAKIT 20
// #define MAX_KELUHAN 51

typedef enum {
    butuhDiagnosa,
    butuhHiperTensi,
    butuhMinumObat,
    butuhPenawar,
    butuhPulang
} StatusPasien;

// unsigned int digunakan untuk variabel yang tidak pernah bernilai negatif
typedef struct {
    int id;
    char penyakit[MAX_PENYAKIT];
    float suhu;
    float beratBadan; // kg
    float tinggiBadan; // cm
    int tekananDarah[2]; // [sistol, diastol]
    unsigned int detakKantung;
    float saturasiOksigen;
    int kadarGulaDarah;
    unsigned int kadarKolesterol;
    unsigned int trombosit;
    // char keluhan[MAX_KELUHAN];
    Obat daftarObat[MAX_OBAT];
    Obat daftarObatResep[MAX_OBAT];
    Stack perutPasien;
    int jumlahObat;
    int jumlahObatResep;
    StatusPasien status;
    int idRuangan;
    int posisiAntrian;
} Pasien;
#endif
```

Gambar 39 Source Code Struktur Data Dokter

```
2 #ifndef DOKTER_H
3 #define DOKTER_H
4
5 #include "../../ADT/header/Queue_Linkedlist.h"
6 #include "../../ADT/header/Ruangan.h"
7
8 #define MAX_PASIEN_RUANGAN 3
9
10 typedef struct Dokter {
11     int id;
12     char username[USERNAME_LEN];
13     int nomorRuang;
14 } Dokter;
15
16 Pasien *assign_pasien_ke_dokter (User *user, Dokter* dokter, Pasien* pasien, ListRuang *ruangan);
17
18 #endif
```

```
✓ #include "dokter.h"
#include "../../ADT/header/Ruangan.h"
#include <stdio.h>
#include <stdlib.h>

✓ Pasien* assign_pasien_ke_dokter(User *user, Dokter* dokter, Pasien* pasien, ListRuang *ruangan) {
    ✓ if (dokter == NULL || pasien == NULL) {
        printf("ERROR: Data dokter atau pasien tidak valid\n");
        return NULL;
    }

    int idx_ruang = dokter->nomorRuang - 1; // ruangan biasanya 1-based
    if (idx_ruang < 0 || idx_ruang >= ruangan->jumlah) {
        printf("ERROR: Dokter belum punya ruangan yang valid\n");
        return NULL;
    }

    // Cek apakah pasien sudah ada di queue
    address current = ruangan->ruang[idx_ruang].Antrian.first;
    while (current != NULL) {
        if (current->pasien->dataPasien == pasien) {
            printf("ERROR: Pasien sudah berada dalam antrian atau ruangan!\n");
            return NULL;
        }
        current = current->next;
    }

    int jumlah_sekarang = queue_size(&ruangan->ruang[idx_ruang].Antrian);
    queue_enqueue(&ruangan->ruang[idx_ruang].Antrian, user);

    // Hitung posisi antrian setelah enqueue
    int jumlah_baru = queue_size(&ruangan->ruang[idx_ruang].Antrian);
    if (jumlah_baru > jumlah_sekarang) {
        if (jumlah_baru <= ruangan->ruang[idx_ruang].kapasitas) {
            pasien->posisiAntrian = 0; // langsung masuk ruangan
        } else {
            pasien->posisiAntrian = jumlah_baru - ruangan->ruang[idx_ruang].kapasitas;
        }
        pasien->idRuang = dokter->nomorRuang; // set id ruangan pasien
        printf(">> Pasien berhasil didaftarkan ke antrian/ruangan\n");
        return pasien;
    }

    printf("ERROR: Gagal mendaftarkan pasien ke antrian\n");
    return NULL;
}
```

Gambar 40 Source Code F07: lihat_user.c

```
char to_lower(char c) {
    if (c >= 'A' && c <= 'Z') return c - 'A' + 'a';
    return c;
}

int compare_nama(char *a, char *b) {
    int i = 0;
    while (a[i] && b[i]) {
        char ca = to_lower(a[i]);
        char cb = to_lower(b[i]);
        if (ca != cb) return (ca < cb) ? -1 : 1;
        i++;
    }
    if (a[i] == b[i]) return 0;
    return a[i] ? 1 : -1;
}

void sort_user (ListUser L, int idx[], int len, int id, int asc){
    for (int i = 0; i < len - 1; i++) {
        for (int j = 0; j < len - i - 1; j++) {
            int cmp;
            if (id) {
                cmp = (L.data[idx[j]].idUser - L.data[idx[j+1]].idUser);
            } else {
                cmp = compare_nama(L.data[idx[j]].username, L.data[idx[j+1]].username);
            }

            if ((asc && cmp > 0) || (!asc && cmp < 0)) {
                int tmp = idx[j];
                idx[j] = idx[j+1];
                idx[j+1] = tmp;
            }
        }
    }
}

void lihat_user (ListUser L) {
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER){
        printf ("\nERROR: Hanya manajer yang bisa melihat user.\n");
        return;
    }

    int pilihan1, pilihan2;
    printf ("Urutkan berdasarkan?\n1. ID\n2. Nama");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan1);
        if (pilihan1 == 1 || pilihan1 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nUrutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan2);
        if (pilihan2 == 1 || pilihan2 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nMenampilkan semua pengguna dengan %s terurut %s...\n",
           (pilihan1 == 1 ? "ID" : "nama"),
           (pilihan1 == 1 ? "ascending" : "descending"));

    int idx[MAX_CAPACITY];
    int len = L.length;
    for (int i = 0; i < len; i++) idx[i] = i;

    sort_user (L, idx, len, pilihan1 == 1, pilihan2 == 1);

    printf("\n-----\n| DAFTAR USER | \n-----\n");
    print_user (L, idx, len, 0);
}
```

```
void lihat_user (ListUser L) {
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER){
        printf ("\nERROR: Hanya manajer yang bisa melihat user.\n");
        return;
    }

    int pilihan1, pilihan2;
    printf ("Urutkan berdasarkan?\n1. ID\n2. Nama");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan1);
        if (pilihan1 == 1 || pilihan1 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nUrutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan2);
        if (pilihan2 == 1 || pilihan2 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nMenampilkan semua pengguna dengan %s terurut %s...\n",
           (pilihan1 == 1 ? "ID" : "nama"),
           (pilihan1 == 1 ? "ascending" : "descending"));

    int idx[MAX_CAPACITY];
    int len = L.length;
    for (int i = 0; i < len; i++) idx[i] = i;

    sort_user (L, idx, len, pilihan1 == 1, pilihan2 == 1);

    printf("\n-----\n| DAFTAR USER | \n-----\n");
    print_user (L, idx, len, 0);
}
```

```
void lihat_pasien (ListUser L){
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER){
        printf ("\nERROR: Hanya manajer yang bisa melihat pasien.\n");
        return;
    }

    int pilihan1, pilihan2;
    printf ("Urutkan berdasarkan?\n1. ID\n2. Nama");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan1);
        if (pilihan1 == 1 || pilihan1 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nUrutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan2);
        if (pilihan2 == 1 || pilihan2 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nMenampilkan semua pengguna dengan %s terurut %s...\n",
           (pilihan1 == 1 ? "ID" : "nama"),
           (pilihan1 == 1 ? "ascending" : "descending"));

    int idx[MAX_CAPACITY];
    int len = L.length;
    for (int i = 0; i < len; i++) idx[i] = i;

    sort_user (L, idx, len, pilihan1 == 1, pilihan2 == 1);

    printf("\n-----\n| DAFTAR PASIEN | \n-----\n");
    print_user (L, idx, len, 1);
}
```

```
void lihat_pasien (ListUser L){
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER){
        printf ("\nERROR: Hanya manajer yang bisa melihat pasien.\n");
        return;
    }

    int pilihan1, pilihan2;
    printf ("Urutkan berdasarkan?\n1. ID\n2. Nama");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan1);
        if (pilihan1 == 1 || pilihan1 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nUrutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan2);
        if (pilihan2 == 1 || pilihan2 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nMenampilkan pasien dengan %s terurut %s...\n",
           (pilihan1 == 1 ? "ID" : "nama"),
           (pilihan1 == 1 ? "ascending" : "descending"));

    int idx[MAX_CAPACITY];
    int len = L.length;
    for (int i = 0; i < len; i++) idx[i] = i;

    sort_user (L, idx, len, pilihan1 == 1, pilihan2 == 1);

    printf("\n-----\n| DAFTAR PASIEN | \n-----\n");
    print_user (L, idx, len, 1);
}
```

```
void lihat_dokter (ListUser L){
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER){
        printf ("\nERROR: Hanya manajer yang bisa melihat dokter.\n");
        return;
    }

    int pilihan1, pilihan2;
    printf ("Urutkan berdasarkan?\n1. ID\n2. Nama");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan1);
        if (pilihan1 == 1 || pilihan1 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nUrutan sort?\n1. ASC (A-Z)\n2. DESC (Z-A)");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan2);
        if (pilihan2 == 1 || pilihan2 == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    printf ("\nMenampilkan dokter dengan %s terurut %s...\n",
           (pilihan1 == 1 ? "ID" : "nama"),
           (pilihan1 == 1 ? "ascending" : "descending"));

    int idx[MAX_CAPACITY];
    int len = L.length;
    for (int i = 0; i < len; i++) idx[i] = i;

    sort_user (L, idx, len, pilihan1 == 1, pilihan2 == 1);

    printf("\n-----\n| DAFTAR DOKTER | \n-----\n");
    print_user (L, idx, len, 2);
}
```

Gambar 41 Source Code F08: cari_user.c

```
void search_penyakit (ListUser L, const char* targetPenyakit){
    int found = 0;
    printf("ID | Nama | Penyakit\n");
    printf("-----\n");

    for (int i = 0; i < L.length; i++) {
        if (L.data[i].role == ROLE_PASIEN && L.data[i].dataPasien != NULL) {
            if (strcmp(L.data[i].dataPasien->penyakit, targetPenyakit) == 0) {
                printf("%-2d | %-15s | %s\n", L.data[i].idUser, L.data[i].username, L.data[i].dataPasien->penyakit);
                found++;
            }
        }
    }

    if (found == 0) {
        printf("Tidak ditemukan pasien dengan penyakit \"%s\".\n", targetPenyakit);
    }
}
```

```
#include "../header/cari_user.h"

int binary_search (ListUser L, int targetElement, int size){
    int left = 0, right = size - 1;

    while (left <= right){
        int middle = (left + right) / 2;

        if (L.data[middle].idUser == targetElement) {
            return middle;
        } else if (L.data[middle].idUser < targetElement) {
            left = middle + 1;
        } else {
            right = middle - 1;
        }
    }
    return -1;
}

int sequential_username (ListUser L, char* targetName, int size){
    for (int i = 0; i < size; i++) {
        if (strcmp(L.data[i].username, targetName) == 0) {
            return i;
        }
    }
    return -1;
}
```

```
void cari_user (ListUser L){
    if (currUser == NULL) {
        printf("ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER) {
        printf("ERROR: Hanya manager yang bisa mencari user.\n");
        return;
    }

    int pilahan, idTarget, userIdx, len = L.length;
    char usernameTarget [USERNAME_LEN];
    printf("Masukkan nama pengguna yang ingin dicari: ");
    scanf(" %[^\n]", usernameTarget);
    printf("CARI USER\n");
    printf("-----\n");
    printf ("Caril berdasarkan ID. IDnya: %s\n", currUser->username);
    while (1) {
        printf("Pilih: ");
        scanf(" %d", &pilahan);
        if (pilahan == 1 || pilahan == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    if (pilahan == 1) {
        printf("Masukkan nomor ID user: ");
        scanf(" %d", &idTarget);
        userIdx = binary_search (L, idTarget, len);

        if (userIdx == -1) {
            printf("Tidak ditemukan pengguna dengan ID user %d.\n", idTarget);
            return;
        }
    } else {
        printf("Masukkan nama user: ");
        scanf(" %[^\n]", usernameTarget);
        userIdx = sequential_username (L, usernameTarget, L.length);
    }
}
```

```
if (userIdx == -1) {
    printf("\nTidak ditemukan pengguna dengan nama '%s'.\n", usernameTarget);
    return;
}

User u = L.data[userIdx];

if (u.role == ROLE_MANAGER) {
    printf("\nID/nama tersebut milik Manager dan tidak ditampilkan.\n");
    return;
}
else {
    const char *role_s = role_to_string(u.role);
    const char *penyakit = (u.role == ROLE_PASIEN && u.dataPasien != NULL) ? u.dataPasien->penyakit : "-";
    printf("ID | Nama | Role | Penyakit\n");
    printf("-----\n");
    printf("%-2d | %-15s | %s | %s\n", u.idUser, u.username, role_s, penyakit);
}
```

```
void cari_pasien (ListUser L){
    if (currUser == NULL) {
        printf("ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER) {
        printf("ERROR: Hanya manager yang bisa mencari pasien.\n");
        return;
    }

    int pilahan, idTarget, userIdx, len = L.length;
    char usernameTarget [USERNAME_LEN];
    printf("Masukkan nama pasien: ");
    printf("CARI PASIEN\n");
    printf("-----\n");
    printf ("Caril berdasarkan ID. IDnya: %s\n", currUser->username);
    while (1) {
        printf("Pilih: ");
        scanf(" %d", &pilahan);
        if (pilahan == 1 || pilahan == 2 || pilahan == 3) break;
        else printf("Pilihan harus 1, 2, atau 3!\n");
    }

    if (pilahan == 3) {
        printf("Masukkan nomor ID pasien: ");
        scanf(" %d", &idTarget);
        userIdx = binary_search (L, idTarget, len);

        if (userIdx == -1) {
            printf("Tidak ditemukan pasien dengan ID %d.\n", idTarget);
            return;
        }
    } else {
        User u = L.data[userIdx];
        if (u.role == ROLE_MANAGER) {
            printf("ID tersebut milik Manager dan tidak ditampilkan.\n");
            return;
        }
    }
}
```

```
else if (u.role == ROLE_PASIEN) {
    printf("User ditemukan, tetapi bukan seorang pasien.\n");
    return;
}
else {
    const char *role_s = role_to_string(u.role);
    const char *penyakit = (u.role == ROLE_PASIEN && u.dataPasien != NULL) ? u.dataPasien->penyakit : "-";
    printf("ID | Nama | Penyakit\n");
    printf("-----\n");
    printf("%-2d | %-15s | %s\n", u.idUser, u.username, penyakit);
}

else if (pilahan == 2) {
    printf("Masukkan nama pasien: ");
    scanf(" %[^\n]", usernameTarget);
    userIdx = sequential_username (L, usernameTarget, L.length);

    if (userIdx == -1) {
        printf("Tidak ditemukan pasien dengan nama '%s'.\n", usernameTarget);
        return;
    }
    else {
        User u = L.data[userIdx];
        if (u.role == ROLE_MANAGER) {
            printf("Nama tersebut milik Manager dan tidak ditampilkan.\n");
            return;
        }
        else if (u.role != ROLE_PASIEN) {
            printf("User ditemukan, tetapi bukan seorang pasien.\n");
            return;
        }
        else {
            const char *role_s = role_to_string(u.role);
            const char *penyakit = (u.role == ROLE_PASIEN && u.dataPasien != NULL) ? u.dataPasien->penyakit : "-";
            printf("ID | Nama | Penyakit\n");
            printf("-----\n");
        }
    }
}
```

```

        }
    } else {
        printf("\n>>> Masukkan nama dokter: ");
        scanf ("%[^\\n]", usernameTarget);
        userIdx = sequential_username(L, usernameTarget, L.length);

        if (userIdx == -1) {
            printf("\nTidak ditemukan dokter dengan nama '%s'.\n", usernameTarget);
            return;
        }
    }

    User u = L.data[userIdx];

    if (u.role == ROLE_MANAGER) {
        printf("\nID tersebut milik Manager dan tidak ditampilkan.\n");
        return;
    } else if (u.role != ROLE_DOKTER) {
        printf("\nUser ditemukan, tetapi bukan seorang dokter.\n");
        return;
    } else {
        const char *role_s = role_to_string(u.role);
        printf("ID | Nama\n-----\n");
        printf("%-2d | %-15s\n", u.idUser, u.username);
    }
}

```

```

void cari_dokter(ListUser L){
    if (currUser == NULL) {
        printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }
    if (currUser->role != ROLE_MANAGER){
        printf ("\nERROR: Hanya manajer yang bisa mencari dokter.\n");
        return;
    }

    int pilihan, idTarget, userIdx, len = L.length;
    char usernameTarget [USERNAME_LEN];
    printf("\n-----+-----+-----+-----+-----+-----+\n");
    printf("          CARI DOKTER          |\n");
    printf("-----+-----+-----+-----+-----+-----+\n");
    printf ("Cari berdasarkan?\n1. ID\n2. Nama");
    while (1) {
        printf("\n>>> Pilihan: ");
        scanf("%d", &pilihan);
        if (pilihan == 1 || pilihan == 2) break;
        else printf("Pilihan harus 1 atau 2!\n");
    }

    if (pilihan == 1){
        printf("\n>>> Masukkan nomor ID dokter: ");
        scanf ("%d", &idTarget);
        userIdx = binary_search (L, idTarget, len);

        if (userIdx == -1){
            printf("\nTidak ditemukan dokter dengan ID %d.\n", idTarget);
            return;
        }
    } else {
        printf("\n>>> Masukkan nama dokter: ");
        scanf ("%[^\\n]", usernameTarget);
        userIdx = sequential_username(L, usernameTarget, L.length);

        if (userIdx == -1) {
            printf("\nTidak ditemukan dokter dengan nama '%s'.\n", usernameTarget);
        }
    }
}

```

Gambar 42 Source Code F09: lihat_semua_antrian.c

```

void lihat_semua_antrian(ListRuangan *ruangan, int num, ListUser *users) {
    if (currUser == NULL) {
        printf("Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.\n");
        return;
    }

    if (currUser->role != ROLE_MANAGER) {
        printf("Anda bukan Manajer, akses ditolak\n");
        return;
    }

    printf("\n-----+-----+-----+-----+-----+-----+\n");
    printf("          SEMUA ANTRIAN          |\n");
    printf("-----+-----+-----+-----+-----+-----+\n");

    // Print room numbers header
    printf("-----+-----+-----+-----+-----+-----+\n");
    for (int i = 0; i < ruangan->jumlah; i++) {
        printf("%d ", ruangan->ruangan[i].nomor);
    }
    printf("\n");
    printf("-----+-----+-----+-----+-----+-----+\n\n");

    // Print details for each room
    for (int i = 0; i < ruangan->jumlah; i++){
        Ruangan r = ruangan->ruangan[i];
        // Skip empty rooms (no doctor assigned)
        if (r.dokter == NULL || strlen(r.dokter->username) == 0) {
            continue;
        }
        printf("... Detail Ruangan %d --\n", ruangan->ruangan[i].nomor);
        printf("Kapasitas : %d\n", r.kapasitas);
        printf("Dokter : Dr. %s\n", r.dokter->username);
        // Print patients in the room (N terdepan dari queue)
        printf("Pasien di dalam ruangan:\n");
        int countPasien = 0;
        Address current = r.Antrian.first;

```

```

        while (current != NULL && countPasien < r.kapasitas) {
            printf(" %d. %s\n", ++countPasien, current->pasien->username);
            current = current->next;
        }
        if (countPasien == 0) {
            printf(" Tidak ada pasien di dalam ruangan saat ini.\n");
        }
        // Print patients in queue (sisanya dari queue)
        printf("Pasien di antrian:\n");
        int counter = 1;
        while (current != NULL) {
            printf(" %d. %s\n", counter++, current->pasien->username);
            current = current->next;
        }
        if (counter == 1) {
            printf(" Tidak ada pasien di antrian saat ini.\n");
        }
        printf("\n");
    }
}

```

Gambar 43 Source Code B06: mainin_antrian.c

```

1 #include "../header/Mainin_Antrian.h"
2
3 void skip_antrian(User *user, ListRuang *ruangan, int banyakUser) {
4     if (currUser == NULL) {
5         printf("\nERROR: Kamu belum login. Silakan login terlebih dahulu\n");
6         return;
7     }
8
9     if (user->role != ROLE_PASIEN) {
10        printf("\nERROR: Hanya pasien yang bisa melewati antrian!\n");
11        return;
12    }
13
14    Pasien *pasien = user->dataPasien;
15    if (pasien->posisiAntrian < 0) {
16        printf("\nERROR: Kamu belum terdaftar dalam antrian check-up.\n");
17        return;
18    }
19
20    // Cek apakah pasien ada di antrian
21    int idxRuang = pasien->idRuang - 1;
22    if (idxRuang < 0 || idxRuang >= ruangan->jumlah) {
23        printf("\nERROR: Ruangan tidak ditemukan!\n");
24        return;
25    }
26
27    AntrianDokter *antrian = &ruangan->ruang[idxRuang].Antrian;
28    int maxDalamRuang = ruangan->ruang[idxRuang].kapasitas; // misal f3
29    address prev = NULL;
30    address curr = antrian->first;
31    int pos = 1;
32    address prevSkip = NULL; // pointer ke node sebelum posisi skip (N)
33    address firstAntrian = NULL; // node ke-(N+1)
34
35    // Cari posisi pasien, dan node ke-(N+1)
36    while (curr != NULL) {
37        if (pos == maxDalamRuang) {
38            | prevSkip = curr;
39            | firstAntrian = curr->next;
40            |
41            if (curr->pasien->dataPasien == pasien) {
42                | break;
43            }
44            prev = curr;
45            curr = curr->next;
46            pos++;
47        }
48
49        if (curr == NULL) {
50            printf("\nERROR: Kamu tidak berada di antrian.\n");
51            return;
52        }
53
54        if (pos <= maxDalamRuang) {
55            printf(">> Kamu sedang berada di dalam ruangan, tidak bisa skip antrian.\n");
56            return;
57        }
58
59        if (curr == firstAntrian) {
60            printf(">> Kamu sudah berada di posisi pertama antrian.\n");
61            return;
62        }
63
64        // Hapus node dari posisi lama
65        if (prev != NULL) {
66            prev->next = curr->next;
67            if (curr == antrian->last) {
68                | antrian->last = prev;
69            }
70
71        }
72
73        // Masukkan ke posisi pertama antrian (setelah pasien di ruangan)
74        if (prevSkip != NULL) {
75            curr->next = prevSkip->next;
76            prevSkip->next = curr;
77            if (curr->next == NULL) {
78                | antrian->last = curr;
79            }
80        }
81
82    }
83
84    printf(">> Kamu berhasil pindah ke posisi pertama antrian!\n");
85 }
```

II. LAMPIRAN HASIL PENGUJIAN PROGRAM

Tabel 4 Hasil Pengujian Program

Fitur	Hasil Pengujian
F01 - Login	<pre>>>> LOGIN +-----+ LOG IN +-----+ Masukan username: GRO Masukan password: NEROIFACANTIK Selamat datang, Pasien GRO! -> Active Pasien: GRO Status: Butuh Didiagnosa</pre>
	<pre>>>> LOGIN ERROR: Anda sudah login sebagai Pasien GRO. Silakan logout terlebih dahulu untuk login dengan akun lain.</pre>
	<pre>>>> LOGIN +-----+ LOG IN +-----+ Masukan username: GRO Masukan password: WRONGPASSWORD ERROR: Password salah untuk pengguna yang bernama GRO!</pre>
	<pre>>>> LOGIN +-----+ LOG IN +-----+ Masukan username: NONEXISTENT Masukan password: WRONGPASSWORD ERROR: Tidak ada Manager, Dokter, atau pun Pasien yang bernama NONEXISTENT!</pre>

Gambar 44 Hasil pengujian F01-Login

F02 - Register

```
>>> REGISTER

+-----+
|                   REGISTRASI               |
+-----+
Username: b
Password: b
>> Pasien b berhasil ditambahkan!

>>> REGISTER

+-----+
|                   REGISTRASI               |
+-----+
Username: GRO

ERROR: Registrasi gagal! Pasien dengan nama GRO sudah terdaftar.

Selamat datang, Manager a!

-> Active Manager: a

>>> REGISTER
Anda sudah login. Silakan logout terlebih dahulu.

>>> REGISTER

+-----+
|                   REGISTRASI               |
+-----+
Username: HAL0

ERROR: Username hanya boleh berisi huruf!
```

Gambar 45 Hasil pengujian F02-Register

F03 - Logout

```
>>> LOGOUT

+-----+
|                   LOG OUT                 |
+-----+

>> Sampai jumpa @a!

>>> LOGOUT
ERROR: Kamu belum login. Silakan login terlebih dahulu dengan command LOGIN.
```

	Gambar 46 Hasil pengujian F03-Logout
F04 - Lupa Password	<pre>>>> LUPA_PASSWORD Username: NONEXISTENT_USER Username tidak terdaftar! >>> REGISTER Anda sudah login. Silakan logout terlebih dahulu. >>> LOGOUT Sampai jumpa Jeffreey! >>> LUPA_PASSWORD Username: NONEXISTENT_USER Username tidak terdaftar!</pre> <pre>>>> LUPA_PASSWORD Username: nimonsslatte Kode Unik: nimonsslatte Kode unik salah!</pre> <pre>>>> LUPA_PASSWORD Username: Neroifa Kode Unik: Neroifa Halo Dokter Neroifa, silakan daftarkan ulang password anda! Password Baru: Neroifa321 Password berhasil diubah!</pre> <pre>>>> LUPA_PASSWORD Username: Neroifa Kode Unik: Neroifa Halo Dokter Neroifa, silakan daftarkan ulang password anda! Password Baru: Neroifa123 Password baru tidak boleh sama dengan password lama!</pre> <p style="text-align: center;">Gambar 47 Hasil pengujian F04-Lupa Password</p>
F05 - Menu & Help	<pre>>>> HELP ===== ===== HELP ===== Kamu belum login sebagai role apapun. Silahkan login terlebih dahulu. LOGIN: Masuk ke dalam akun yang sudah terdaftar REGISTER: Membuat akun baru Footnote: Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar Jangan lupa untuk memasukkan input yang valid</pre>

```
>>> HELP

===== HELP =====

Halo Dokter Neroifa. Kamu memanggil command HELP. Kamu pasti sedang kebingungan.
Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

LOGOUT: Keluar dari akun yang sedang digunakan
DIAGNOSIS: Melakukan diagnosis penyakit pasien berdasarkan kondisi tubuh pasien

Footnote:
Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
Jangan lupa untuk memasukkan input yang valid

>>> LOGOUT
Sampai jumpa Neroifa!

>>> LOGIN
Masukan username: GRO
Masukan password: NeroifaCantik
Selamat pagi Pasien GRO! Ada keluhan apa ?

>>> HELP

===== HELP =====

Halo Pasien GRO. Kamu memanggil command HELP. Kamu pasti sedang kebingungan.
Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

LOGOUT: Keluar dari akun yang sedang digunakan
DAFTAR_CHECKUP: Mendaftarkan diri untuk pemeriksaan dokter

Footnote:
Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
Jangan lupa untuk memasukkan input yang valid

>>> HELP

===== HELP =====

Halo Manager nimonsslatte. Kenapa kamu memanggil command HELP? Kan kamu manager, tapi yasudahlah kamu pasti sedang kebingungan.
Berikut adalah hal-hal yang dapat kamu lakukan sekarang:

LOGOUT: Keluar dari akun yang sedang digunakan
TAMBAH_DOKTER: Mendaftarkan dokter baru ke sistem

Footnote:
Untuk menggunakan aplikasi, silahkan masukkan nama fungsi yang terdaftar
Jangan lupa untuk memasukkan input yang valid
```

Gambar 48 Hasil pengujian F05-menu dan help

F06 - Lihat Denah & Ruangan

```
>>> LIHAT_DENAH  
+-----+-----+-----+-----+-----+-----+-----+  
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |  
+-----+-----+-----+-----+-----+
```

```
>>> LIHAT_RUANGAN  
Masukkan nomor ruangan (1-8): 1
```

```
>> DETAIL RUANGAN 1  
Kapasitas : 3  
Dokter : Dr. NEROIFA  
Pasien di dalam ruangan:  
1. GRO  
2. nimonganteng  
Pasien di antrian:  
Tidak ada pasien di antrian saat ini.  
-----
```

```
>>> LIHAT_RUANGAN  
Masukkan nomor ruangan (1-8): 9  
Nomor ruangan tidak valid.
```

Gambar 49 Hasil pengujian F06-Lihat Denah & Ruangan

F07 - Lihat User

```
>>> LIHAT_USER  
Urutkan berdasarkan?  
1. ID  
2. Nama  
>>> Pilihan: 1
```

```
Urutan sort?  
1. ASC (A-Z)  
2. DESC (Z-A)  
>>> Pilihan: 1
```

Menampilkan semua pengguna dengan ID terurut ascending...

DAFTAR USER			
ID	Nama	Role	Penyakit
1	NEROIFA	Dokter	-
2	GRO	Pasien	
3	nimonsganteng	Pasien	

```
>>> LIHAT_PASIEN  
Urutkan berdasarkan?  
1. ID  
2. Nama  
>>> Pilihan: 1
```

```
Urutan sort?  
1. ASC (A-Z)  
2. DESC (Z-A)  
>>> Pilihan: 1
```

Menampilkan pasien dengan ID terurut ascending...

DAFTAR PASIEN		
ID	Nama	Penyakit
2	GRO	
3	nimonsganteng	

```
>>> LIHAT_DOKTER
Urutkan berdasarkan?
1. ID
2. Nama
>>> Pilihan: 2

Urutan sort?
1. ASC (A-Z)
2. DESC (Z-A)
>>> Pilihan: 1

Menampilkan dokter dengan nama terurut descending...

+-----+
|                   DAFTAR DOKTER               |
+-----+
| ID | Nama
+-----+
| 4  | HALO
| 1  | NEROIFA
```

Gambar 50 Hasil pengujian F07- Lihat User

F08 - Cari user

```
>>> CARI_USER

+-----+
|                   CARI USER |
+-----+
Cari berdasarkan?
1. ID
2. Nama
>>> Pilihan: 1

>>> Masukkan nomor ID user: 2
ID | Nama             | Role       | Penyakit
+-----+
2  | GRO              | Pasien    |
```

```
>>> CARI_DOKTER

+-----+
|                   CARI DOKTER |
+-----+
Cari berdasarkan?
1. ID
2. Nama
>>> Pilihan: 1

>>> Masukkan nomor ID dokter: 1
ID | Nama
+-----+
1  | NEROIFA
```

```
>>> CARI_DOKTER

+-----+
|                   CARI DOKTER |
+-----+
Cari berdasarkan?
1. ID
2. Nama
>>> Pilihan: 2

>>> Masukkan nama dokter: GRO

User ditemukan, tetapi bukan seorang dokter.
```

	Gambar 51 Hasil pengujian F08- Cari user
F09 - Lihat Antrian	<pre>>>> LIHAT_SEMUA_ANTRIAN +-----+ SEMUA ANTRIAN +-----+ +---+---+---+---+---+---+---+ 1 2 3 4 5 6 7 8 +---+---+---+---+---+---+---+---+ --- Detail Ruangan 1 --- Kapasitas : 3 Dokter : Dr. NEROIFA Pasien di dalam ruangan: 1. GRO 2. nimonganteng Pasien di antrian: Tidak ada pasien di antrian saat ini. --- Detail Ruangan 5 --- Kapasitas : 3 Dokter : Dr. HALO Pasien di dalam ruangan: Tidak ada pasien di dalam ruangan saat ini. Pasien di antrian: Tidak ada pasien di antrian saat ini.</pre>
F10 - Tambah & Assign Dokter	<pre>>>> ASSIGN_DOKTER Username: Neroifa Ruangan: 1 Dokter Neroifa berhasil diassign ke ruangan 1! >>> LIHAT_RUANGAN 1 Masukkan nomor ruangan (1-8): --- Detail Ruangan 1 --- Kapasitas : 3 Dokter : Neroifa Pasien di dalam ruangan: Tidak ada pasien di dalam ruangan saat ini. -----</pre>

```
>>> TAMBAH_DOKTER
Username: newdoctor
Password: 123
Dokter newdoctor berhasil ditambahkan!

>>> ASSIGN_DOKTER
Username: newdoctor
Ruangan: 2
Dokter newdoctor berhasil diassign ke ruangan 2!
```

```
>>> ASSIGN_DOKTER
Username: newdoctor
Ruangan: 1
Dokter newdoctor sudah diassign ke ruangan 2!
```

Gambar 53 Hasil pengujian F10

F11 - Diagnosis

```
>>> DIAGNOSIS
+-----+
|          DIAGNOSA          |
+-----+
[dr. NEROIFA] Pasien @GRO terdiagnosa mengidap penyakit: Infeksi Parah
[dr. NEROIFA] Jangan lupa untuk diobatin ya!
HELP: Untuk mengobati pasien @GRO, ketik NGOBATIN!
```

```
>>> DIAGNOSIS
[dr. NEROIFA] Pasien @GRO udah didiagnosa, tinggal dikasih obat aja.
```

Gambar 54 Hasil pengujian F11 - Diagnosis

F12 - Ngobatin

```
>>> NGOBATIN
+-----+
|          PENGOBATAN          |
+-----+
[dr. NEROIFA] Resep untuk @GRO (Penyakit: Infeksi Parah):
1. Amoxicillin
2. Vitamin C
3. Ibuprofen
4. Amlodipine
```

	<pre>>>> NGOBATIN</pre> <pre>+-----+ PENGOBATAN +-----+</pre> <p>[dr. NEROIFA] Pasien @GRO sudah diberi obat, tidak perlu diobati lagi.</p>
F13 - Aku Boleh Pulang Gak Dok	<pre>>>> PULANG</pre> <pre>+-----+ PULANG +-----+</pre> <p>Kamu belum menghabiskan seluruh obat yang diberikan. Yuk minum obatmu dulu!</p> <pre>>>> PULANG</pre> <pre>+-----+ PULANG +-----+</pre> <p>>> Dokter sedang memeriksa...</p> <p>Maaf, kamu kayaknya masih belum bisa pulang, masih sakit ya? Urutan obat yang diharapkan: Amoxicillin -> Vitamin C -> Ibuprofen -> Amlodipine Urutan obat yang kamu minum: Amoxicillin -> Vitamin C -> Amlodipine -> Ibuprofen Kunjungi dokter untuk meminta penawar yang sesuai yaa!</p>

Gambar 55 Hasil pengujian F12 - Ngobatin

Gambar 56 Hasil pengujian F13

F14 - Daftar-Check-up

```
>>> DAFTAR_CHECKUP

+-----+
|          DAFTAR CHECK UP          |
+-----+
>> Input data medis:
Suhu tubuh (Celcius): 30
Tekanan darah (sistol/diastol, contoh: 120 80): 116 78
Detak jantung (bpm): 88
Saturasi oksigen (persentase): 98
Kadar gula darah (mg/dL): 43
Berat badan (kg): 70
Tinggi badan (cm): 180
Kadar kolestrol (mg/dL): 32
Trombosit (ribu/µL): 12

Berikut adalah daftar dokter yang tersedia:
1. dr. NEROIFA (Ruangan 1) - Total Ruangan + Antrian: 2
2. dr. HALO (Ruangan 5) - Total Ruangan + Antrian: 0

Pilih dokter (1-2): 1
>> Pasien berhasil didaftarkan ke antrian/ruangan
>> Pasien berhasil didaftarkan ke ruangan 1.

>> Pendaftaran berhasil! <<
[@g] Kamu terdaftar pada antrian dr. NEROIFA di ruangan 1.
[@g] Posisi antrian Kamu: 0
```

Gambar 57 Hasil pengujian F14 - Daftar Check-Up

F15 - Antrian Saya!

```
>>> LIHAT_ANTRIAN

+-----+
|          STATUS ANTRIAN          |
+-----+
[@GRO] Kamu lagi di ruangan dokter, loh. Dokternya ga lagi tidur kan?

>>> LIHAT_ANTRIAN

+-----+
|          STATUS ANTRIAN          |
+-----+
[@R] Status antrian Anda:
Dokter: dr. NEROIFA
Ruangan: 1
Posisi antrian: 1 dari 2
```

```
-> Active Pasien: T  
Status: Butuh Didiagnosa
```

```
>>> LIHAT_ANTRIAN
```

```
+-----+  
| STATUS ANTRIAN |  
+-----+  
[@T] Status antrian Anda:  
Dokter: dr. NEROIFA  
Ruangan: 1  
Posisi antrian: 2 dari 2
```

Gambar 58 Hasil pengujian F15 - Antrian Saya!

F16 - Minum Obat

```
>>> MINUM_OBAT
```

```
+-----+  
| MINUM OBAT |  
+-----+  
>> Yuk minum obatmu dulu..  
[@GRO] Kamu punya 4 obat yang harus diminum.
```

```
===== DAFTAR OBAT =====
```

```
Urutan minum obat sesuai resep dokter:
```

```
Amoxicillin -> Vitamin C -> Ibuprofen -> Amlodipine
```

Obat yang harus diminum:

1. Amoxicillin
2. Vitamin C
3. Ibuprofen
4. Amlodipine

Obat yang sudah diminum:

Belum ada obat yang diminum.

Pilih obat untuk diminum: 1

GLEKGLEKGLEK.. Amoxicillin berhasil diminum!

```

=====
 DAFTAR OBAT =====
 Urutan minum obat sesuai resep dokter:
 Amoxicillin -> Vitamin C -> Ibuprofen -> Amlodipine

 Obat yang harus diminum:
 1. Ibuprofen
 2. Amlodipine

 Obat yang sudah diminum:
 Amoxicillin -> Vitamin C

 Pilih obat untuk diminum: 2
 GLEKGLEKGLEK.. Amlodipine berhasil diminum!
 Kamu merasa makin gak enak badan... kayaknya kamu salah minum obat deh..
 Dok! Butuh obat penawar!! X o X

```

Gambar 59 Hasil pengujian F16

F17 - Minum Penawar	<pre> >>> MINUM_PENAWAR +-----+ MINUM PENAWAR +-----+ ERROR: Uwekkk!!! Ibuprofen keluar dan kembali ke inventory ERROR: Urutan obat masih belum sesuai resep dokter! Muntahkan lagi satu obat... ERROR: Uwekkk!!! Amlodipine keluar dan kembali ke inventory >> Berhasil! Urutan obat terdepan sudah sesuai resep dokter! </pre>
---------------------	---

Gambar 60 Hasil pengujian F17

F18 - Exit	<pre> >>> EXIT Apakah Anda mau keluar dari rumah sakit? (y/n) Pilihan Anda: N Kembali ke menu utama... >>> EXIT Apakah Anda mau keluar dari rumah sakit? (y/n) Pilihan Anda: n Kembali ke menu utama... >>> EXIT Apakah Anda mau keluar dari rumah sakit? (y/n) Pilihan Anda: Y User keluar dari Rumah Sakit, Sampai jumpa! 192:tubes alpro 2025 wahyuindragunawan\$ █ </pre>
------------	--

```
>>> LOGIN
Masukan username: GRO
Masukan password: NeroifaCantik
Selamat pagi Pasien GRO! Ada keluhan apa ?

>>> EXIT

Apakah Anda mau keluar dari rumah sakit? (y/n)
Pilihan Anda: n

Kembali ke menu utama...

>>> EXIT

Apakah Anda mau keluar dari rumah sakit? (y/n)
Pilihan Anda: y

GRO keluar dari Rumah Sakit,
Sampai jumpa!
192:tubes alpro 2025 wahyuindragunawan$ █
```

Gambar 61 Hasil pengujian F18

B05 - Dead or Alive

```
-> Active Pasien: GRO
Status: Butuh Minum Obat
Sisa Nyawa:3
```

Urutan minum obat sesuai resep dokter:
Amoxicillin -> Vitamin C -> Ibuprofen -> Amlodipine

Obat yang harus diminum:
1. Amoxicillin
2. Vitamin C
3. Ibuprofen
4. Amlodipine

Obat yang sudah diminum:
Belum ada obat yang diminum.

Pilih obat untuk diminum: 2
GLEKGLEKGLEK.. Vitamin C berhasil diminum!
Kamu merasa makin gak enak badan... kayaknya kamu salah minum obat deh..
Dok! Butuh obat penawar!! X o X

```
-> Active Pasien: GRO
Status: Butuh Minum Penawar
Sisa Nyawa:2
```

```

>>> MINUM_PENAWAR

+-----+
|          MINUM PENAWAR          |
+-----+
Sisa nyawamu: 2

ERROR: Uwekkk!!! Vitamin C keluar dan kembali ke inventory
>> Berhasil! Urutan obat terdepan sudah sesuai resep dokter!

-> Active Pasien: GRO
    Status: Butuh Minum Obat
    Sisa Nyawa:3

Obat yang sudah diminum:
Ibuprofen -> Amoxicillin

Pilih obat untuk diminum: 2
GLEKGLEKGLEK.. Vitamin C berhasil diminum!
Kamu merasa makin gak enak badan... kayaknya kamu salah minum obat deh..
Dok! Butuh obat penawar!! X o X

-> Active Pasien: GRO
    Status: Pasien Sudah Meninggal
    Sisa Nyawa:0
Game over.. Pasien dinyatakan Ded dan dikeluarkan dari rumah sakit. RIP o7

```

Gambar 62 Hasil pengujian B05

B06 - Mainin Antrian

```

+-----+
|          STATUS ANTRIAN          |
+-----+
[@F] Status antrian Anda:
    Dokter: dr. NEROIFA
    Ruangan: 1
    Posisi antrian: 4 dari 4

-> Active Pasien: F
    Status: Butuh Didiagnosa
    Sisa Nyawa:3

```

```
>>> SKIP_ANTRIAN
>> Kamu berhasil pindah ke posisi pertama antrian!

-> Active Pasien: F
    Status: Butuh Didiagnosa
    Sisa Nyawa:3

>>> LIHAT_ANTRIAN

+-----+
|                      STATUS ANTRIAN          |
+-----+
[@F] Status antrian Anda:
    Dokter: dr. NEROIFA
    Ruangan: 1
    Posisi antrian: 1 dari 4

-> Active Pasien: F
    Status: Butuh Didiagnosa
    Sisa Nyawa:3
```

Gambar 63 Hasil pengujian B06

III. LAMPIRAN ASISTENSI

**Form MoM Asistensi Tugas Besar
IF1210/Algoritma dan Pemrograman 1
Sem. 2 2024/2025**

Nomor Asistensi : 1
No. Kelompok/Kelas : A/K03
Tanggal asistensi : 2-Mei-2025

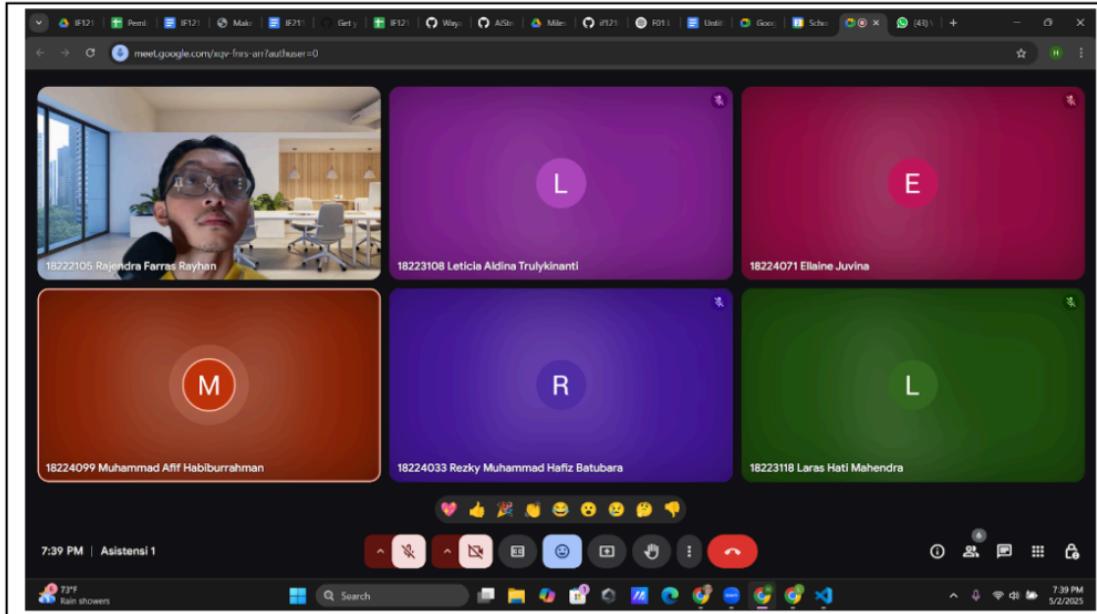
Anggota kelompok	NIM / Nama (Hanya yang Hadir)
1	18223108 / Leticia Aldina Trulykinanti
2	18223118 / Laras Hati Mahendra
3	18224033 / Rezky Muhammad Hafiz Batubara
4	18224071 / Ellaine Juvina
5	18224099 / Muhammad Afif Habiburrahman
Asisten pembimbing	NIM / Nama
	18222105 / Rajendra Farris Rayhan

Catatan Asistensi:

Rangkuman Diskusi
<ol style="list-style-type: none">Penulisan header harus disesuaikan terkait alur output-inputnya data dan jangan lupa buat makefile agar bisa di run langsung.Struktur file sebaiknya modular, data tidak disimpan secara hardcoded di satu file saja.ADT (Abstract Data Type) disarankan disimpan dalam folder terpisah seperti adt/.Fungsi gabungan diperbolehkan, contoh: antrian bisa pakai map, queue, dan linked list.Milestone 1: F1–F6, F10, F18 (login, register, logout, password check, cari user, exit point).Minimal progress 40% harus dicapai sebelum deadline (11 Mei 2025).Laporan harus pakai ringkasan fungsi yang dijalankan.
Tindak Lanjut
<ol style="list-style-type: none">Sesuaikan isi header dengan fungsionalitas modul terkait (misalnya: login.h, register.h, dll).Pisahkan data seperti user, obat, dll ke dalam file .h masing-masing (user.h, obat.h).Buat folder adt/ yang berisi file list.h, stack.h, queue.h, dll untuk struktur modular.Buat dokumentasi pemanggilan antar ADT agar referensi jelas dan terstruktur.Fokus implementasi fitur-fitur dasar dan buat dokumentasi per fitur yang selesai.Membuat tracker progress/sheets <u>A-K03 Pembagian Tugas</u>Siapkan dokumen laporan dan mulai isi deskripsi fungsi yang telah dibuat.
Dokumentasi

Gambar 64 Form asistensi pertama

**Form MoM Asistensi Tugas Besar
IF1210/Algoritma dan Pemrograman 1
Sem. 2 2024/2025**



Gambar 65 Form asistensi pertama

Form MoM Asistensi Tugas Besar
IF1210/Algoritma dan Pemrograman 1
Sem. 2 2024/2025

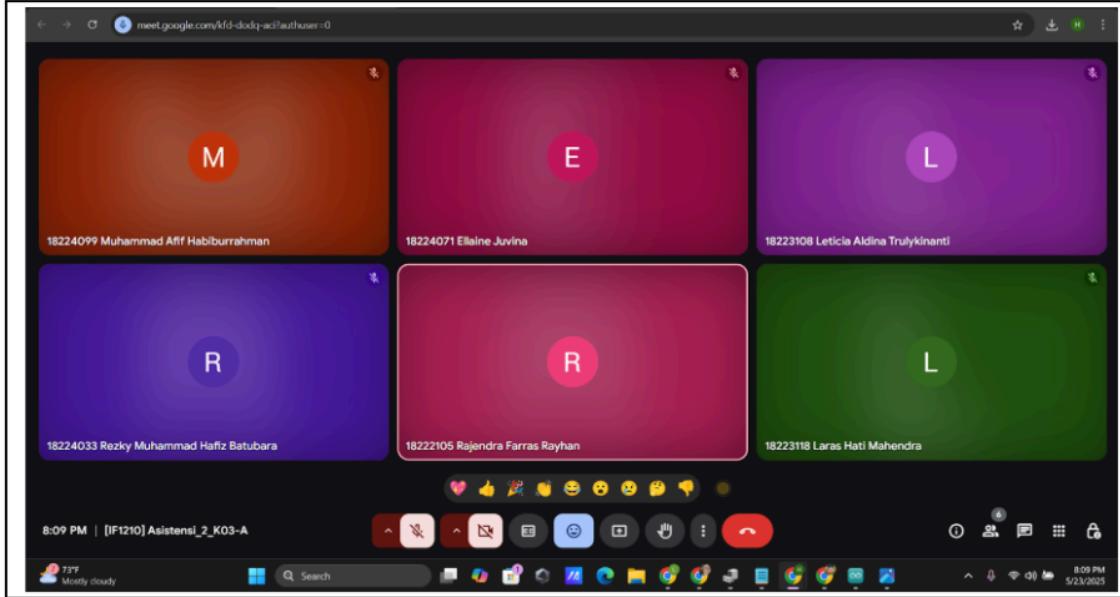
Nomor Asistensi	:	2																		
No. Kelompok/Kelas	:	A/K03																		
Tanggal asistensi	:	23-Mei-2025																		
Anggota kelompok	<table border="1"> <thead> <tr> <th colspan="2">NIM / Nama (Hanya yang Hadir)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>18223108 / Leticia Aldina Trulykinanti</td> </tr> <tr> <td>2</td> <td>18223118 / Laras Hati Mahendra</td> </tr> <tr> <td>3</td> <td>18224033 / Rezky Muhammad Hafiz Batubara</td> </tr> <tr> <td>4</td> <td>18224071 / Ellaine Juvina</td> </tr> <tr> <td>5</td> <td>18224099 / Muhammad Afif Habiburrrahman</td> </tr> <tr> <td colspan="2"> </td> </tr> <tr> <td colspan="2">NIM / Nama</td> </tr> <tr> <td colspan="2">18222105 / Rajendra Farris Rayhan</td> </tr> </tbody> </table>		NIM / Nama (Hanya yang Hadir)		1	18223108 / Leticia Aldina Trulykinanti	2	18223118 / Laras Hati Mahendra	3	18224033 / Rezky Muhammad Hafiz Batubara	4	18224071 / Ellaine Juvina	5	18224099 / Muhammad Afif Habiburrrahman			NIM / Nama		18222105 / Rajendra Farris Rayhan	
NIM / Nama (Hanya yang Hadir)																				
1	18223108 / Leticia Aldina Trulykinanti																			
2	18223118 / Laras Hati Mahendra																			
3	18224033 / Rezky Muhammad Hafiz Batubara																			
4	18224071 / Ellaine Juvina																			
5	18224099 / Muhammad Afif Habiburrrahman																			
NIM / Nama																				
18222105 / Rajendra Farris Rayhan																				
Asisten pembimbing																				

Catatan Asistensi:

Rangkuman Diskusi
<ol style="list-style-type: none"> Setiap user hanya memiliki satu peran (role), yaitu pasien, dokter, atau manajer, dan hanya menyimpan data yang sesuai dengan perannya. Untuk mengelola data per role agar lebih rapi dan tidak saling tumpang tindih, disarankan menggunakan union atau tabel terpisah sesuai role. Fungsi <u>lihatUser()</u> dapat diarahkan langsung ke fungsi spesifik berdasarkan role, seperti <u>lihatPasien()</u>, <u>lihatDokter()</u>, atau <u>lihatManajer()</u>. Penamaan fungsi perlu menggunakan awalan (prefix) yang mencerminkan modul atau role-nya, agar mudah dikenali dan menghindari konflik nama. Akses terhadap data juga harus dibatasi sesuai role. Contohnya, dokter hanya dapat melihat data pasien yang memang berelasi dengannya.
Tindak Lanjut
<ol style="list-style-type: none"> Perlu dilakukan refactor struktur data user, baik dengan union maupun dengan relasi antar tabel (untuk SQL). Pisahkan fungsi-fungsi tampilan data sesuai dengan role, seperti <u>lihatPasien()</u>, <u>lihatDokter()</u>, dan <u>lihatManajer()</u>, agar alur kerja lebih jelas. Terapkan penamaan fungsi dengan format yang konsisten, misalnya <MODUL>_<STRUKTUR>_<AKSI>, untuk mencegah konflik nama saat kompilasi. Implementasikan sistem Role-Based Access Control (RBAC) di level aplikasi agar setiap role hanya bisa mengakses data yang sesuai dengan haknya. Lakukan uji coba login dan akses data berdasarkan role untuk memastikan alur kerja berjalan.
Dokumentasi

Gambar 66 Form asistensi kedua

**Form MoM Asistensi Tugas Besar
IF1210/Algoritma dan Pemrograman 1
Sem. 2 2024/2025**



Gambar 67 Form asistensi kedua