

# **APLIKASI PENDATAAN PASIEN RAWAT INAP RUMAH SAKIT**



**Hendri Darmawan**

**1210 181 044**

**2 D4 Teknik Telekomunikasi B**

**PRODI TEKNIK TELEKOMUNIKASI  
POLITEKNIKELEKTRONIKAELEKTRONIKANEGERI  
SURABAYA  
2019**

## KATA PENGANTAR

Dengan memanjatkan puji syukur kehadiran Tuhan Yang Maha Esa, karena dengan karuniaNya saya dapat menyelesaikan laporan *final project* aplikasi untuk Linux yang berjudul Aplikasi Pendataan Pasien Rawat Inap Rumah Sakit dengan tepat waktu. Tidak lupa juga saya mengucapkan terima kasih atas dukungan dari :

1. Pak Akuwan Saleh sebagai dosen pembimbing mata kuliah Bengkel Pemrograman Shell semester III.
2. Orang tua dan teman-teman 2 D4 Telkom B yang telah membantu dalam penyelesaian laporan *final project* ini.
3. Semua pihak yang turut serta membantu.

Dengan menyadari segala keterbatasan yang ada maka saya sangat berterima kasih apabila ada pihak-pihak yang berkenan memberikan kritik dan saran untuk laporan *final project* ini.

Surabaya, 10 Desember 2019

Penulis

## DAFTAR ISI

HALAMAN JUDUL .....	i
KATA PENGANTAR .....	2
DAFTAR ISI .....	3
BAB I PENDAHULUAN	
1.1 Tujuan .....	4
BAB II Landasan Teori	
2.1 Pengkondisian .....	4
2.2 Perulangan .....	7
2.3 Array.....	11
2.4 Subrutin atau Fungsi .....	13
2.5 <i>Graphical User Interface (GUI)</i> .....	17
BAB III ANALISIS DATA PENGUJIAN	
3.1 Source Code Data .....	21
3.2 Analisis Data .....	33
BAB IV PENUTUP	
4.1 Kesimpulan .....	64
DAFTAR PUSTAKA .....	65

# BAB I

## PENDAHULUAN

### 1.1 Tujuan

- 1.1.1 Mengaplikasikan pemrograman shell dalam Aplikasi Pendataan Pasien Rawat Inap Rumah Sakit dalam tampilan GUI
- 1.1.2 Memudahkan pendataan administrasi rumah sakit sehingga meminimalisir kesalahan atau *human error*

# BAB II

## LANDASAN TEORI

### 2.1 Pengkondisian

#### KONSTRUKSI *if*

Statement builtin *if* berfungsi untuk melakukan seleksi berdasarkan suatu kondisi tertentu. Secara umum ada dua bentuk umum sintak perintah *if*, seperti ditunjukkan dibawah ini

***Sintak :***

1.     ***if [ kondisi ]***  
          ***then***  
                  *statements*  
          ***fi***
2.     ***if [ kondisi ]***  
          ***then***

```

        statements
    else
        statements
fi

```

Perbedaan antara kedua bentuk adalah bentuk pertama mempunyai perintah tunggal jika ekspresi/kondisi yang diuji benar, sedangkan bentuk kedua mempunyai banyak perintah yang akan dijalankan jika ekspresi yang diuji benar.

Contoh bentuk pertama:

```

let hasil = "$b * $c"
if [ "$hasil" = 10 ]
then
    echo "Hasil perkalian kedua bilangan = $hasil"
fi

```

Contoh bentuk kedua:

```

let hasil = "$b * $c"
if [ "$hasil" = 10 ]
then
    echo "Hasil perkalian kedua bilangan = $hasil"
else
    echo "selesai"
fi

```

Kalau diperhatikan, perintah *if* ini hampir sama dengan perintah *if* pada bahasa-bahasa tingkat tinggi, seperti Pascal, C, dan juga hampir sama dengan perintah *if* pada batch file-nya DOS. Pada bentuk pertama maupun bentuk kedua dari sintak diatas adalah statement dalam blok *if..fi* akan dieksekusi apabila kondisi *if* terpenuhi. Dari kedua bentuk diatas dapat pula ditambahkan perintah

untuk pengecekan kondisi dengan *elif* (else if), contoh sintaknya adalah sebagai berikut:

```
3.  if [ kondisi ];  
    then  
        perintah1;  
    elif [ kondisi2 ];  
    then  
        perintah2;  
    else  
        alternatif_perintah;  
    fi
```

klausa *else* akan dieksekusi jika *if* tidak terpenuhi, sebaliknya jika *if* terpenuhi maka *else* tidak akan dieksekusi.

Contoh bentuk ketiga:

```
if winter  
    then  
        snowremoval  
        weatherstrip  
elif spring  
    then  
        startgarden  
        mowlawn  
else  
        echo Something is wrong  
fi
```

## KONSTRUKSI case

Case digunakan untuk menyederhanakan pemakaian *if* yang berantai, sehingga dengan case, kondisi dapat dikelompokkan secara logis dengan lebih jelas dan mudah untuk ditulis. Statement **case** juga digunakan untuk menyeleksi kondisi majemuk, dibanding **if**, pemakaian case lebih efisien.

**Sintak :**

```
case string in
    pilihan)
        commands
    ;;
    pilihan)
        commands
    ;;
*)
    default commands
;;
esac
```

Case diakhiri dengan **esac** dan pada setiap kelompok instruksi diakhiri dengan **;;**. Pada akhir pilihan yaitu **\*)** yang berarti adalah “default”, bila kondisi tidak memenuhi pola sebelumnya. Contoh:

```
let hasil = "$b * $c"
case $hasil in
    10)
        echo "Hasil perkalian kedua bilangan = $hasil"
    ;;
    *)
        echo "Selesai"
esac
```

## 2.2 Perulangan

### KONSTRUKSI FOR

For digunakan untuk pengulangan dengan menggunakan variabel (*name*) yang pada setiap pengulangan akan diganti dengan nilai yang berada pada daftar (*list = word1 word2 ...*). Sintak dari perintah FOR adalah sebagai berikut:

#### Sintak 1.

```
for name in word1 word2 ...  
do  
    do-list  
done
```

Contoh:

```
#!/bin/bash  
for buah in apel jeruk mangga salak  
do  
    echo $buah adalah buah  
done
```

Pada contoh program diatas variabel *\$buah* akan diganti dengan data pada list yaitu apel, jeruk, mangga dan salak.

#### Sintak 2.

```
for name  
do  
    do-list  
done
```

Contoh:

```
#!/bin/bash  
for var  
do  
    echo $var  
done
```



*\$.for2 satu 2 tiga*

Contoh program menggunakan sintak2 variabel *\$var* akan diganti dengan data hasil pembacaan argument (satu, 2, tiga) yang disertakan saat script dijalankan.

## KONSTRUKSI WHILE

While digunakan untuk pengulangan instruksi, yang umumnya dibatasi dengan suatu kondisi. Selama kondisi tersebut TRUE, maka pengulangan terus dilakukan. Loop akan berhenti, bila kondisi FALSE, atau program keluar dari blok while melalui `exit` atau `break`. Sintak dari perintah WHILE adalah sebagai berikut:

### Sintak 1. While – end

```
while ( test_condition )  
  
    commands /kumpulan perintah  
  
end
```

Contoh:

```
set i=$#argv  
while ($i)  
    echo -n $argv[$i]  
    @i--  
end
```

Pada contoh program menggunakan sintak1 akan mencetak parameter yang diterima oleh program, tetapi dalam susunan terbalik karena nilai variabel "*i*" dikurangi satu persatu dimulai dari nilai yang tertinggi. Perintah ***echo -n*** digunakan agar setiap kali menampilkan satu parameter, parameter berikutnya tidak akan tercetak pada baris berikutnya.

### Sintak 2. While – do

```
while [ test_condition ]  
  
do
```

*commands*

***done***

Contoh:

```
i=1;  
while [ $i -le 10 ];  
do  
    echo "$i,";  
    let i=$i+2;  
done
```

Contoh program dengan sintak2 diatas menunjukkan kondisi tidak terpenuhi pada saat nilai i=11 (9+2), sehingga perintah dalam blok while tidak dieksekusi lagi dan nilai i=11 tidak pernah ditampilkan pada layar.

## INSTRUKSI DUMMY

Instruksi dummy adalah instruksi yang tidak melakukan apa-apa, namun instruksi ini memberikan status exit 0 (TRUE). Oleh karena itu, instruksi dummy dapat digunakan sebagai kondisi forever pada loop (misalnya ***while***).

Simbol instruksi dummy adalah → :

## KONSTRUKSI UNTIL

Jika while akan mengulang selama kondisi benar, lain halnya dengan statement until yang akan mengulang selama kondisi salah, berikut contoh script menggunakan ***until***

<b>Sintak</b>	<b><i>until condition</i></b>
	<b><i>do</i></b>
	<i>list</i>
	<b><i>done</i></b>

Contoh:

```
i=1;
until [ $i -gt 10 ];
do
    echo "$i,";
    let i=$i+1;
done
```

perhatikan kondisi until yang salah [ \$i -gt 10], dimana nilai awal i=1 dan akan berhenti apabila nilai i = 11 (bernilai benar) 11 -gt 10.

## KONSTRUKSI SELECT

Select berguna untuk pembuatan layout berbentuk menu pilihan, sewaktu dijalankan bash akan menampilkan daftar menu yang diambil dari item list.

### Sintak

```
select varname in (item list)
do
    commands
done
```

## 2.3 Array

### ARRAY

Array adalah kumpulan variabel dengan tipe sejenis, dimana array ini merupakan feature Bash yang cukup menarik. Array juga merupakan salah satu hal yang cukup penting dalam bahasa pemrograman, bahwa bisa dibayangkan array ini sebagai tumpukan buku - buku dimeja belajar. Inisialisasi array sebagai berikut:

```
Array=(element1 element2 ... elementN)
```

Contoh

```
buah=(Melon,Apel,Durian);  
echo ${buah[*]};
```

Dari contoh diatas bisa dilihat bahwa membuat tipe array di Bash begitu mudah, secara otomatis array *buah* diciptakan dan *string Melon* menempati index pertama dari array buah, perlu diketahui bahwa array di Bash dimulai dari *index 0*, jadi array buah mempunyai struktur seperti berikut:

buah[0] berisi Melon

buah[1] berisi Apel

buah[2] berisi Durian

0,1,2 adalah index array, berarti ada 3 elemen pada array buah, untuk menampilkan isi semua elemen array digunakan perintah substitusi seperti pada contoh diatas, dengan index berisi "\*" atau "@". dengan adanya index array tentunya user dapat mengisi array perindexnya dan menampilkan isi array sesuai dengan index yang diinginkan. Lihat contoh berikut:

```
bulan[0]=31  
bulan[1]=28  
bulan[2]=31  
bulan[3]=30  
echo "Banyak hari bulan pebruari= ${bulan[2]} hari"
```

cara lain yaitu dengan mendeklarasikan array secara eksplisit menggunakan statement *declare*. Contohnya:

```
declare -a myarray
```

mendeklarasikan variabel *myarray* sebagai array dengan opsi *-a*, kemudian dapat memberinya nilai baik untuk semua elemen atau hanya elemen tertentu saja dengan perintah perulangan pengisian elemen array dapat lebih dipermudah, lihat contoh :

```

declare -a angka
i=0;
while [ $i -le 4 ];
do
let isi=$i*2;
angka[$i]=$isi;
let i=$i+1;
done

#tampilkan semua elemen array
#dengan indexnya berisi "*" atau "@"
echo ${angka[*]};

```

Dari kedua contoh array diatas, dapat dijelaskan bahwa array dapat digunakan untuk operasi string (string operations) dan perintah substitusi (command substitution).

## 2.4 Subrutin atau Fungsi

### Subrutin atau Fungsi

Merupakan bagian script atau program yang berisi kumpulan beberapa statement yang melaksanakan tugas tertentu. Dengan subrutin kode script tentunya akan lebih sederhana dan terstruktur, karena sekali fungsi telah dibuat dan berhasil maka akan dapat digunakan kapan saja bila diinginkan. Beberapa hal mengenai fungsi ini adalah:

- Memungkinkan penyusunan kode script ke dalam bentuk modul-modul kecil yang lebih efisien dengan tugasnya masing-masing.
- Mencegah penulisan kode yang berulang - ulang.

Di dalam shell fungsi juga bisa didefinisikan interaktif maupun secara script program, dan meskipun didefinisikan secara interaktif, sebuah fungsi juga bisa dipanggil melalui script yang dibuat dalam sebuah file dengan catatan fungsi tersebut sudah di export. Setelah melalui mekanisme export ini sub-shell juga bisa

memanggil fungsi tersebut. Jadi fungsi adalah program yang dapat dipanggil oleh program lainnya dengan menggunakan notasi NamaFungsi(). Fungsi memberikan exit status (\$?) yang dinyatakan dengan *return nr*, atau nilai 0 sebagai default. Untuk membuat subrutin shell telah menyediakan keyword *function* seperti pada bahasa C, akan tetapi ini bersifat optional (artinya boleh digunakan boleh tidak). Bentuk umum dalam mendefinisikan fungsi dalam bash shell adalah sebagai berikut:

**Sintak:**

```
Nama_fungsi () { command; command; }  
Function nama_fungsi { command; command; }  
Function nama_fungsi () { command; command; }
```

nama\_fungsi adalah pengenalan (identifier) yang aturan penamaannya sama seperti pemberian nama variabel, setelah fungsi dideklarasikan atau dibuat dapat dipanggil dengan menyebutkan nama fungsinya. lebih jelasnya lihat contoh script berikut:

```
#!/bin/bash  
function hai_hello() {  
    echo "Hello, apa khabar"  
}  
  
#panggil fungsi  
hai_hello;
```

jika keyword *function* disertakan maka boleh tidak menggunakan tanda kurung (), tetapi jika keyword *function* tidak disertakan maka tanda kurung harus digunakan, lihat contoh berikut:

```
function hai_hello{  
    echo "Hello,apa khabar"  
}
```

```

balas(){
    echo "Baik-baik saja";
    echo "Bagaimana dengan anda ?";
}

```

#### ❖ Mengirim argumen sebagai parameter ke fungsi

Tentunya suatu fungsi lebih berdaya guna apabila dapat menerima argumen yang dikirim oleh pemanggilnya dan memproses argumen tersebut didalam fungsinya, fungsi yang telah dibuat pada bash shell tentunya harus dapat melakukan hal tersebut, apabila pada pemanggilan fungsi disertakan argumen untuk diproses fungsi tersebut, maka bash akan menyimpan argumen - argumen tersebut pada parameter posisi 1,2,3,dan seterusnya..., dengan memanfaatkan parameter posisi tersebut tentunya dapat diambil nilai yang telah dikirim. lebih jelasnya lihat contoh berikut:

```

#!/bin/bash

function hello{
    if [ -z $1 ]; then
        echo "Hello, apa khabar anda"
    else
        echo "Hello $1, apa khabar";
    fi
}

#masukkan nama anda disini
echo -n "Nama anda :";
read nama

#panggil fungsi dan kirim isi variabel nama ke fungsi untuk dicetak
hello $nama;

```

perhatikan fungsi hello, sebelum mencetak pesan terlebih dahulu melakukan pemeriksaan dengan *if* terhadap parameter posisi \$1 apabila kosong maka pesan "Hello, apa khabar anda" yang akan ditampilkan, tetapi jika ada string yang di

inputkan maka string tersebut akan dicetak di dalam blok *else* pada fungsi. argumen pertama diteruskan ke variabel 1, argumen kedua pada variabel 2, dan seterusnya ... jika argumen yang dikirim lebih dari satu.

### ❖ Cakupan Variabel

Secara default variabel - variabel yang digunakan dalam script adalah variabel bersifat global, maksud global adalah bahwa variabel tersebut dikenal dan dapat diakses oleh semua fungsi dalam script, tetapi bash menyediakan keyword *local* yang berfungsi membatasi cakupan (scope) suatu variabel agar dikenal hanya oleh fungsi yang mendeklarasikannya. Jadi variabel dapat didefinisikan dalam fungsi sebagai variable local atau global. Hal yang perlu diperhatikan, nama variable yang digunakan dalam sebuah fungsi, jangan sampai bentrok dengan nama variable yang sama di luar fungsi, sehingga tidak terjadi isi variable berubah. Perhatikan contoh berikut:

```
#!/bin/bash
proses(){
    echo "Isi variabel a=$a";
}
a=2;
proses();
proses $a
```

Jika ditambahkan *local a* pada fungsi proses menjadi

```
proses(){
    local a;
    echo -e "a didalam fungsi, a=$a";
}
a=10;
proses()
echo "a diluar fungsi, a=$a"
proses $a
```



Pada contoh diatas jelas perbedaannya jika mendeklarasikan variabel memakai keyword *local* menyebabkan variabel tersebut hanya berlaku pada fungsi yang mendeklarasikannya. Pada contoh dalam fungsi proses variabel *a* dideklarasikan sebagai *variabel local* dan tidak diberi nilai.

## 2.5 Graphical User Interface (GUI)

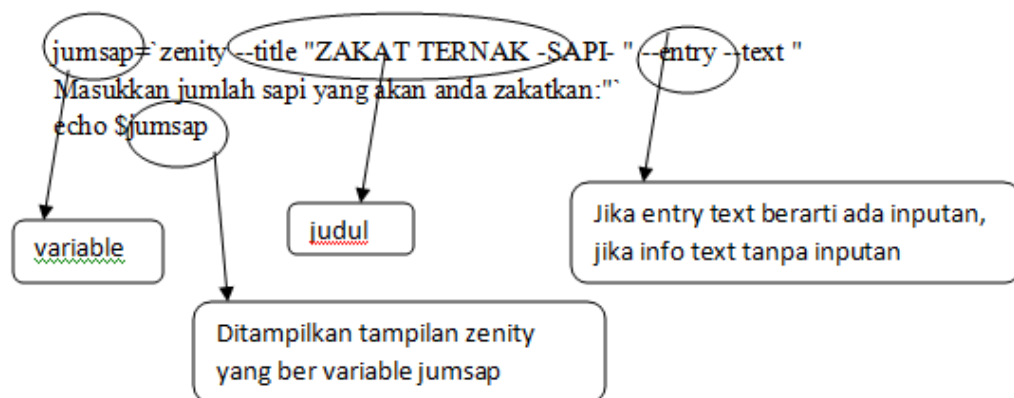
### Zenity

Zenity mempunyai peran sebagai pembuat GUI pada shell programming atau yang disebut *Grafik User Interface* dimana mempunyai fungsi mempermudah user untuk mengoperasikan programnya melalui grafis secara interaktif. Zenity mempunyai banyak opsi seperti:

- entry : meminta input dari keyboard
- info : menampilkan text yang dimana berfungsi sebagai sebuah info
- list : membuat list berdasarkan column dan row secara tertabel dan lain-lain.

Dengan opsi seperti diatas zenity dapat digunakan untuk membuat sebuah *question dialog box*. Disamping itu zenity juga dapat digunakan untuk aplikasi yang lain seperti *calendar*, *entry*, *error*, *info*, *file selection*, *list*, *notification*, *progress*, *warning*, *scale* dan *text info*. Pada bab ini akan di ilustrasikan bagaimana membuat aplikasi zenity dialog.

Contoh sintak zenity :



## Kalender Dialog

Dengan menggunakan option `--calendar` dapat dibuat sebuah kalender dialog. Pada kalender dialog ini user diijinkan untuk memilih inisial yang khusus pada perintah date yaitu dengan menggunakan option :

`--text=text`

Spesifikasi “text” digunakan untuk menampilkan teks pada kalender dialog.

`--day=day`

Spesifikasi “day” digunakan untuk memilih tanggal di dalam kalender dialog.

Day harus diberi nilai angka untuk tanggal dari 1 sampai 31.

`--month=month`

Spesifikasi “month” digunakan untuk memilih bulan di dalam kalender dialog.

Month harus diberi nilai untuk bulan dari 1 sampai 12.

`--year=year`

Spesifikasi “year” digunakan untuk memilih tahun di dalam kalender dialog.

`--date-format=format`

Menentukan kembali format dari dialog kalender setelah seleksi tanggal (date). Format standar tergantung pada system lokal dan format yang dapat diterima oleh fungsi *strftime* misalnya %A %d/%m/%y.

## File Selection Dialog

Untuk membuat *file selection dialog* digunakan opsi `--file-selection`. *Zenity* akan melakukan seleksi file atau direktori ke output standard. Mode default *file selection dialog* adalah buka file. *File selection dialog* memiliki beberapa opsi:

`--filename=filename`

Menentukan file atau direktori yang dipilih pada dialog pemilihan file ketika dialog yang pertama ditampilkan.

`--multiple`

Menentukan pemilihan beberapa nama file dalam dialog pemilihan file.

`--directory`

Menentukan pilihan direktori pada dialog pemilihan file.

*--save*

Set dialog pemilihan file ke mode save.

*--separator=separator*

Menentukan string yang digunakan untuk membagi kembali daftar nama file.

## List Dialog

Menggunakan opsi *--list* Untuk membuat *list dialog*. Zenity akan mengembalikan entry dalam kolom pertama baris teks yang dipilih ke output standar. Data untuk dialog kolom harus ditentukan menurut kolom, baris demi baris. Data dapat disediakan untuk dialog melalui input standar. Setiap entri harus dipisahkan oleh karakter baris baru. Jika menggunakan opsi *--checkboxlist* atau *--radiolist*, setiap baris harus dimulai dengan 'TRUE' atau 'FALSE'. List dialog memiliki opsi berikut:

*--column=column*

Menentukan header kolom yang ditampilkan dalam list dialog. Harus ditentukan opsi *--column* untuk setiap kolom yang ingin ditampilkan dalam dialog.

*--checkboxlist*

Menentukan bahwa kolom pertama pada list dialog berisi kotak cek.

*--radiolist*

Menentukan bahwa kolom pertama pada list dialog berisi kotak radio.

*--editable*

Memungkinkan ditampilkan item yang akan diedit.

*--separator=separator*

Menentukan string apa yang digunakan ketika dialog mengembalikan daftar entry yang dipilih.

*--print-column=column*

Menentukan apakah kolom harus dicetak pada seleksi. Default kolom adalah '1'. 'ALL' yang dapat digunakan untuk encetak semua kolom dalam list.

Untuk melihat penjelasan lebih detail tentang penggunaan zenity, dapat dilihat di terminal dengan mengetik ***man zenity***, maka akan muncul petunjuk penggunaan zenity.

## BAB III

### ANALISIS DATA PENGUJIAN

#### 3.1 Source Code Data

DataRS.sh

```
#!/bin/bash
```

```
admin=$(zenity --forms --title="Login" --text="Masukkan akun administrator" \
```

```
--add-entry="Nama Pengguna" \
```

```
--add-password="Kata Sandi")
```

```
IFS="|" read -r user pwod <<< "$admin" ;
```

```
if [ $? -eq 1 ];then
```

```
exit 0;
```

```
fi
```

```
if [[ $pwod == "sehat" || $pwod == "waras" && $user != "" ]];then
```

```
(
```

```
echo "10" ; sleep 0.5
```

```
echo "# Verifikasi akun" ; sleep 1
```

```
echo "35" ; sleep 1
```

```
echo "# Mengupdate data" ; sleep 1
```

```
echo "75" ; sleep 2
```

```
echo "# Proses berhasil" ; sleep 0.5
```

```
echo "100" ; sleep 1
```

```
) | zenity --progress \
```

```
--title="Silahkan Tunggu" \
```

```
--percentage=0 --auto-close --width=300
```

```
if [ $? -eq 1 ];then
```

```
zenity --error --width=200 --height=70 \
```

```
--text="Proses berhenti!"
```

```
exit 0;
```

```

fi
let i=1;
let bou=20;
let ala=20;
let ang=20;
let se=30;
let ro=30;
let si=30;
let seme=50;
let wili=50;
let brom=50;
declare -A dokter;
declare -A perawat;
declare -A tensi;
declare -A suhu;
declare -A tglp;
declare -A periksa;
sis() {
zenity --list --height=260 --width=483 --title="Data Ketersediaan Kamar" --
text="Pastikan terdapat kamar kosong untuk pasien" \
--column="Kelas 1" --column="Kelas 2" --column="Kelas 3" "Bougenville=$bou
tersisa" "Sedudo=$se tersisa" "Semeru=$seme tersisa" "Alamanda=$ala tersisa"
"Roro Kuning=$ro tersisa" "Wilis=$wili tersisa" "Anggrek=$ang tersisa"
"Singokromo=$si tersisa" "Bromo=$brom tersisa"
if [ $? -eq 1 ]; then
home;
else
kelas;
fi
}
kelas() {

```

```

kelas=$(zenity --list --title="Pilih Kelas Kamar" --text="<b>Tarif kamar :</b>
\nKelas 1 = Rp 3.000.000/hari\nKelas 2 = Rp 1.500.000/hari\nKelas 3 = Rp
500.000/hari" \
--radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE "Kelas 1"
FALSE "Kelas 2" FALSE "Kelas 3")
if [ $? -eq 1 ]; then
    sisa;
else kamar;
fi
}
kamar () {
if [[ $kelas == "Kelas 1" ]]; then
kmr=$(zenity --list --title="Kamar Kelas 1 Tersedia" --text="Tarif kamar kelas 1
Rp 3000.000/hari" \
--radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE
"Bougenville" FALSE "Alamanda" FALSE "Anggrek")
let kali[i]=3000000
IFS="|" read -r kamarr[i] <<< "$kmr" ;
if [[ ${kamarr[i]} == "Bougenville" && $bou > 0 ]];then
kamar[i]={kamarr[i]}$bou
let bou=$bou-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Alamanda" && $sala > 0 ]];then
kamar[i]={kamarr[i]}$sala
let ala=$sala-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Anggrek" && $sang > 0 ]];then
kamar[i]={kamarr[i]}$sang
let ang=$sang-1

```

```

zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
else
kamarr[i]=""
zenity --error --title="Maaf" --text "Kesalahan! Silahkan lihat kamar yang tersedia"
--width=200 --height=70
sis;
fi
elif [[ $kelas == "Kelas 2" ]]; then
kmr=$(zenity --list --title="Kamar Kelas 2 Tersedia" --text="Tarif kamar kelas 2
Rp 1.500.000/hari" \
--radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE "Sedudo"
FALSE "Roro Kuning" FALSE "Singokromo")
let kali[i]=1500000
IFS="|" read -r kamarr[i] <<< "$kmr" ;
if [[ ${kamarr[i]} == "Sedudo" && $se > 0 ]];then
kamar[i]=${kamarr[i]} $se
let se=$se-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Roro Kuning" && $ro > 0 ]];then
kamar[i]=${kamarr[i]} $ro
let ro=$ro-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Singokromo" && $si > 0 ]];then
kamar[i]=${kamarr[i]} $si
let si=$si-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
else
kamarr[i]=""

```



```

zenity --error --title="Maaf" --text "Kesalahan! Silahkan lihat kamar yang tersedia"
--width=200 --height=70
fi
elif [[ $kelas == "Kelas 3" ]]; then
kmr=$(zenity --list --title="Kamar Kelas 3 Tersedia" --text="Tarif kamar kelas 3
Rp 500.000/hari" \
--radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE "Semeru"
FALSE "Wilis" FALSE "Bromo")
let kali[i]=500000
IFS="|" read -r kamarr[i] <<< "$kmr" ;
if [[ ${kamarr[i]} == "Semeru" && $seme > 0 ]];then
kamar[i]={kamarr[i]}$seme
let seme=$seme-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Wilis" && $wili > 0 ]];then
kamar[i]={kamarr[i]}$wili
let wili=$wili-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Bromo" && $brom > 0 ]];then
kamar[i]={kamarr[i]}$brom
let brom=$brom-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
else
kamarr[i]=""
zenity --error --title="Maaf" --text "Kesalahan! Silahkan lihat kamar yang tersedia"
--width=200 --height=70
fi
fi
}

```

```

main() {
if [[ $fitur == "Pelayanan" ]]; then
    pasien=$(zenity --list --title="Jenis Pelayanan" --text="Pilih jenis pelayanan pada
pasien" \
        --radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE
        "Menginap" FALSE "Merawat" FALSE "Membayar")
    if [ $pasien == "Menginap" ]; then
        datapasien=$(zenity --forms --title="Data Pasien" --text="Masukkan data pasien"
        \
            --add-entry="Nama Pasien" \
            --add-combo="Jenis Kelamin" --combo-values="Laki-Laki|Perempuan"\
            --add-entry="Usia" \
            --add-entry="Alamat" \
            --add-entry="Keluhan" \
            --add-calendar="Tanggal")
        IFS="|" read -r nama[i] jenis[i] usia[i] alamat[i] keluhan[i] d <<< "$datapasien" ;
        tanggal[i]="20${d:6:2}-${d:3:2}-${d:0:2}"
        no[i]=$i;
        dok[i]=0
        per[i]=0
        status[i]="Dirawat"
        x[i]=0
        if [[ ${nama[i]} != "" ]]; then
            sisa;
        else
            zenity --error --title="Galat" --text "Lengkapi data pasien!" --width=200 --
            height=70
        fi
        elif [ $pasien == "Merawat" ]; then
            rekampasien=$(zenity --forms --title="Data Pasien" --text="Masukkan data
pasien" \
                --add-entry="Nomor Urut Pasien" \

```

```

--add-calendar="Tanggal Periksa" \
--add-entry="Dokter"\
--add-entry="Perawat"\
--add-entry="Tekanan Darah"\
--add-entry="Suhu Badan")
IFS="|" read -r noo1 d dokter1 perawat1 tensi1 suhu1 <<< "$rekampasien" ;
tglp1="20${d:6:2}-${d:3:2}-${d:0:2}"
if [ $noo1 == ${no[$noo1]} ]; then
x[$noo1]=$((x[$noo1]+1))
if [[ $dokter1 != "" ]]; then
noo[$noo1]=$noo1                                dokter[$noo1,${x[$noo1]}]=$dokter1
perawat[$noo1,${x[$noo1]}]=$perawat1            tglp[$noo1,${x[$noo1]}]=$tglp1
tensi[$noo1,${x[$noo1]}]=$tensi1  suhu[$noo1,${x[$noo1]}]=$suhu1
dok[$noo1]=$((dok[$noo1]+500000))
periksa[$noo1,${x[$noo1]}]="Ke ${x[$noo1]}"
zenity --info --title="Sukses" --text "Rekam medik pasien oleh dokter telah
tersimpan" --width=200 --height=70
elif [[ $perawat1 != "" ]]; then
noo[$noo1]=$noo1                                dokter[$noo1,${x[$noo1]}]=$dokter1
perawat[$noo1,${x[$noo1]}]=$perawat1            tglp[$noo1,${x[$noo1]}]=$tglp1
tensi[$noo1,${x[$noo1]}]=$tensi1  suhu[$noo1,${x[$noo1]}]=$suhu1
per[$noo1]=$((per[$noo1]+250000))
periksa[$noo1,${x[$noo1]}]="Ke ${x[$noo1]}"
zenity --info --title="Sukses" --text "Rekam medik pasien oleh perawat telah
tersimpan" --width=200 --height=70
fi
else
zenity --error --title="Galat" --text "Data pasien tidak ditemukan" --width=200 --
height=70
fi
elif [ $pasien == "Membayar" ]; then

```

```

biayapasien=$(zenity --forms --title="Pembayaran Tagihan" --text="Masukkan
data pasien" \
--add-entry="Nomor Urut Pasien" \
--add-calendar="Tanggal Keluar" )
IFS="|" read -r noo1 d <<< "$biayapasien" ;
tglk[$noo1]="20${d:6:2}-${d:3:2}-${d:0:2}"
if [ $noo1 == ${no[$noo1]} ]; then
keluar=$(date -d "${tglk[$noo1]}" "+%s")
masuk=$(date -d "${tanggal[$noo1]}" "+%s")
let hari=$(( ( $keluar/86400 ) - ( $masuk/86400 ) )+1)
let kamar=(${kali[$noo1]}*$hari)
let penanganan=(${dok[$noo1]}+${per[$noo1]})
let biaya=($kamar+$penanganan)
zenity --info --title="Total Bayar" --text "<b>Rincian biaya :</b>\nKamar =
$skamar \nPenanganan = $penanganan\n<b>Total yang harus dibayar = $biaya</b>"
--width=500 --height=70
no[$noo1]=" "
status[$noo1]="Keluar"
if [[ ${kamarr[$noo1]} == "Bougenville" ]];then
let bou=$bou+1;
elif [[ ${kamarr[$noo1]} == "Alamanda" ]];then
let ala=$ala+1;
elif [[ ${kamarr[$noo1]} == "Anggrek" ]];then
let ang=$ang+1;
elif [[ ${kamarr[$noo1]} == "Sedudo" ]];then
let se=$se+1;
elif [[ ${kamarr[$noo1]} == "Roro Kuning" ]];then
let ro=$ro+1;
elif [[ ${kamarr[$noo1]} == "Singokromo" ]];then
let si=$si+1;
elif [[ ${kamarr[$noo1]} == "Semeru" ]];then
let seme=$seme+1;

```

```

elif [[ ${kamarr[$noo1]} == "Wilis" ]];then
let wili=$wili+1;
elif [[ ${kamarr[$noo1]} == "Bromo" ]];then
let brom=$brom+1;
fi
echo "${no[$noo1]}
${nama[$noo1]}|${jenis[$noo1]}|${usia[$noo1]}|${alamat[$noo1]}|${keluhan[$
noo1]}|${tanggal[$noo1]}|${tglk[$noo1]}|Shari|$biaya" >>
/home/hdr/snap/dataRS
else
zenity --error --title="Galat" --text "Pasien tidak terdaftar" --width=200 --height=70
fi
fi
elif [[ $fitur == "Rekam Medik" ]]; then
for o in $(seq 1 $i)
do
for v in $(seq 1 ${x[o]})
do
if [ ${noo[o]} != "" ]; then
if [ ${noo[o]} == ${no[o]} ];then
echo ${noo[o]}
echo ${nama[o]}
echo ${tglp[$o,$v]}
echo ${dokter[$o,$v]}
echo ${perawat[$o,$v]}
echo ${tensi[$o,$v]}
echo ${suhu[$o,$v]}
echo ${kamar[o]}
echo ${periksa[$o,$v]}
fi
fi
done

```

```

done | zenity --list --height=720 --width=1366 --title="Data Pasien Rumah Sakit"
--text="Data pasien menginap Rumah Sakit"\
--column="No." --column="Nama" --column="Tanggal Periksa" --
column="Dokter" --column="Perawat" --column="Tekanan Darah" --
column="Suhu Badan" --column="Kamar Pasien" --column="Periksa"
elif [ $fitur == "Database" ]; then
for o in $(seq 1 $i)
do
if [[ ${nama[o]} != "" && ${kamar[o]} != "" ]]; then
masuk=$(date -d "${tanggal[o]}" "+%s")
now=$(date "+%s")
let hari=$(( ( $now/86400 ) - ( $masuk/86400 ) ))
let penanganan[o]=${dok[o]}+${per[o]}
let biaya[o]=(${kali[o]}*$hari+${penanganan[o]})
echo ${no[o]}
echo ${nama[o]}
echo ${jenis[o]}
echo ${usia[o]}
echo ${alamat[o]}
echo ${keluhan[o]}
echo ${tanggal[o]}
echo ${kamar[o]}
echo ${biaya[o]}
echo ${status[o]}
fi
done | zenity --list --height=720 --width=1366 --title="Data Pasien Rumah Sakit"
--text="Data pasien menginap Rumah Sakit"\
--column="No." --column="Nama" --column="Jenis Kelamin" --column="Usia" -
-column="Alamat" --column="Keluhan" --column="Tanggal menginap" --
column="Kamar Pasien" --column="Tagihan" --column="Status"
elif [[ $fitur == "Edit Database" ]]; then
datap pasien=$(zenity --forms --title="Data Pasien" --text="Masukkan data pasien" \

```

```

--add-entry="Nomor Urut Pasien" \
--add-entry="Nama Pasien" \
--add-combo="Jenis Kelamin" --combo-values="Laki-Laki|Perempuan"\
--add-entry="Usia" \
--add-entry="Alamat" \
--add-entry="Keluhan" \
--add-calendar="Tanggal")

IFS="|" read -r noo1 nama1 jenis1 usia1 alamat1 keluhan1 d <<< "$datap pasien" ;
tanggal="20${d:6:2}-${d:3:2}-${d:0:2}"
if [[ $noo1 == ${no[$noo1]} && $noo1 != "" ]];then
no[$noo1]=$noo1      nama[$noo1]=${nama1}      jenis[$noo1]=${jenis1}
usia[$noo1]=${usia1}  alamat[$noo1]=${alamat1}  keluhan[$noo1]=${keluhan1}
tanggal[$noo1]=${tanggal}
zenity --info --title="Sukses" --text "Data pasien telah diperbaiki" --width=200 --
height=70
elif [ $? -eq 1 ];then
zenity --error --title="Galat" --text "Tidak ada data yang berubah" --width=200 --
height=70
else
zenity --error --title="Galat" --text "Data pasien tidak ditemukan" --width=200 --
height=70
fi
elif [ $fitur == "History" ]; then
cat /home/hdr/snap/dataRS | \
awk -F '|' '{
print NR;
for(i=1;i<=NF;i++){
print $i;
}
}' | \
zenity --list --height=720 --width=1366 --title="History Rawat Inap" --text="Data
pasien yang pernah menginap di Rumah Sakit"\

```

```

--column="No." --column="Nama" --column="Jenis Kelamin" --column="Usia" -
--column="Alamat" --column="Keluhan" --column="Tanggal Masuk" --
column="Tanggal Keluar" --column="Lama Menginap" --column="Biaya"
elif [ $fitur == "Logout" ]; then
zenity --question --text="Apakah anda yakin ingin logout?" --default-cancel --
width=200 --height=70
if [ $? -eq 0 ];then
pwod2=$(zenity --forms --width=350 --title="Masukkan Password" --
text="Masukkan password untuk melanjutkan" \
--add-password="Kata Sandi")
if [[ $pwod2 == $pwod ]];then
zenity --info --title="OK" --text "Akun dengan username $user telah logout!" --
width=200 --height=70
exit 0;
else zenity --error --title="Galat" --text "Masukkan password dengan benar!" --
width=200 --height=70
fi
fi
fi
}
home() {
fitur=$(zenity --forms --title="Pilih Fitur" --text="Ini adalah fitur dari aplikasi"\
--add-combo="Fitur Aplikasi" --combo-values="Pelayanan|Rekam
Medik|Database|Edit Database|History|Logout")
if [ $? -eq 1 ]; then
zenity --info --title="Jangan Keluar!!!" --text "Sistem harus selalu dalam kondisi
aktif 24 jam" --width=200 --height=70
fi
main;
}
while 1>0
do

```



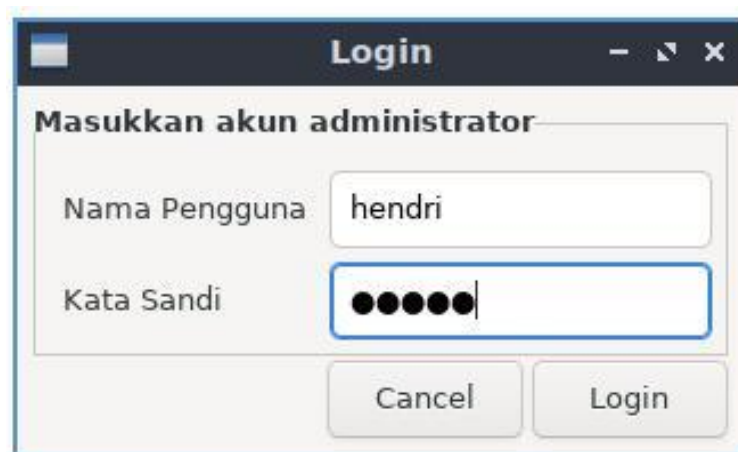
```

home;
if [[ ${nama[i]} != "" && ${kamarr[i]} != "" ]]; then
let i=$i+1;
fi
done
else zenity --error --title="Gagal Login" --text "Masukkan akun yang benar!" --
width=200 --height=70
exit 0;
fi

```

### 3.2 Analisis Data

Pada *final project* Pemrograman Shell ini dibuat program Pendataan Pasien Rawat Inap Rumah Sakit. GUI aplikasi ini dibuat menggunakan zenity serta menggunakan database untuk penyimpanan permanen. Untuk menjalankan program ini dengan menggunakan perintah `./DataRS.sh` dengan mengubah perizinan file terlebih dahulu agar program dapat dijalankan atau dapat menggunakan perintah `bash DataRS.sh`. Pada saat program pertama kali dijalankan yaitu terdapat halaman login sebelum memasuki aplikasi dengan tujuan keamanan, tampilan dari halaman login seperti di bawah ini



Source code:

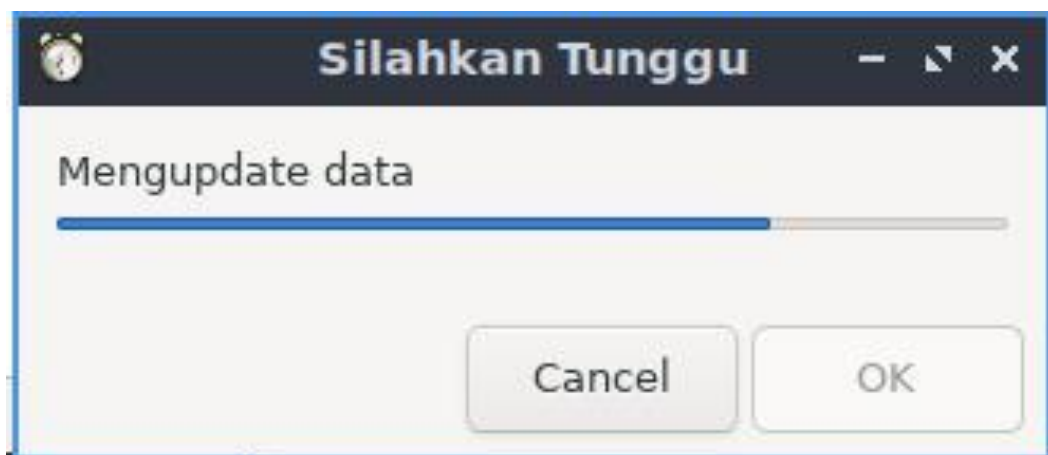
```

admin=$(zenity --forms --title="Login" --text="Masukkan akun administrator" \
--add-entry="Nama Pengguna" \

```

```
--add-password="Kata Sandi")
IFS="|" read -r user pwod <<< "$admin" ;
if [ $? -eq 1 ];then
exit 0;
fi
```

Pada source code telah ditentukan kata sandi yang valid untuk mengakses aplikasi diantaranya "waras" atau "sehat". Tidak ada ketentuan untuk input nama pengguna karena ini hanya sebagai identitas user yang memakai aplikasi ini. Apabila login berhasil maka program akan muncul tampilan memuat dan user dipersilahkan untuk menunggu beberapa detik. Tampilan memuat setelah login berhasil tersebut seperti di bawah ini

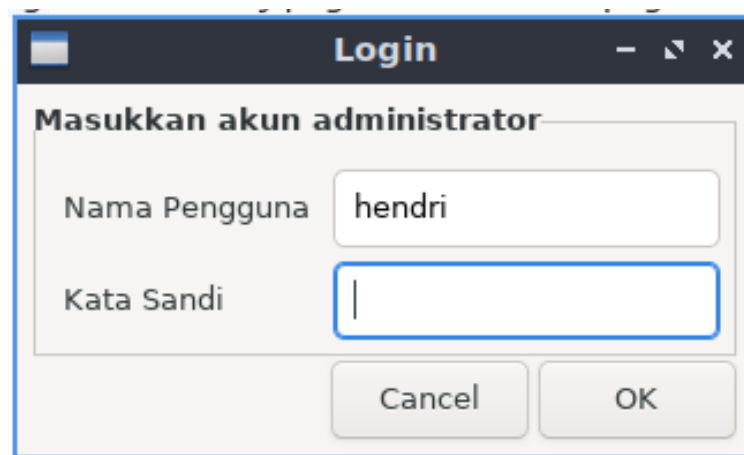


Source code:

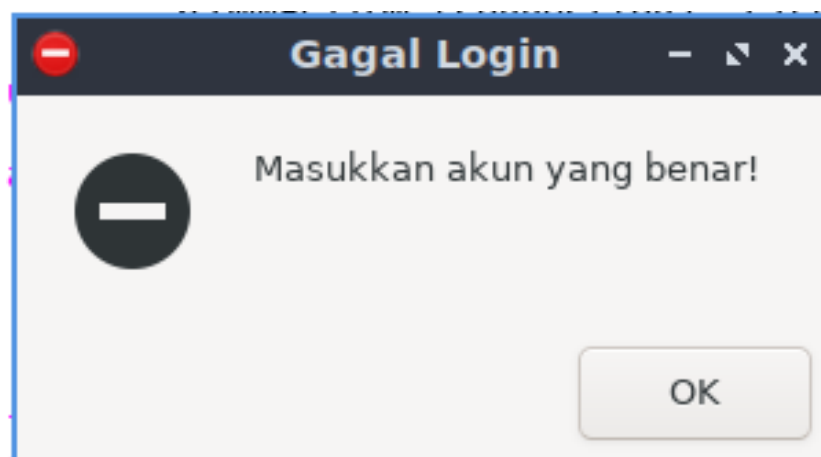
```
if [[ $pwod == "sehat" || $pwod == "waras" && $user != "" ]];then
(
echo "10" ; sleep 0.5
echo "# Verifikasi akun" ; sleep 1
echo "35" ; sleep 1
echo "# Mengupdate data" ; sleep 1
echo "75" ; sleep 2
echo "# Proses berhasil" ; sleep 0.5
echo "100" ; sleep 1
```

```
) | zenity --progress \  
--title="Silahkan Tunggu" \  
--percentage=0 --auto-close --width=300
```

Apabila kata sandi yang dimasukkan salah atau tidak diketikkan kata sandi ataupun kolom nama pengguna tidak diketikkan maka seperti gambar di bawah ini



maka program akan memunculkan peringatan gagal login lalu program berhenti, seperti gambar di bawah ini

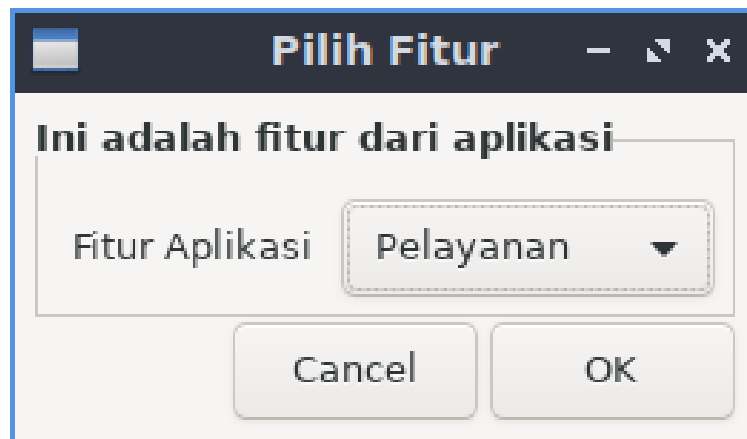


Hal di atas karena source code berikut

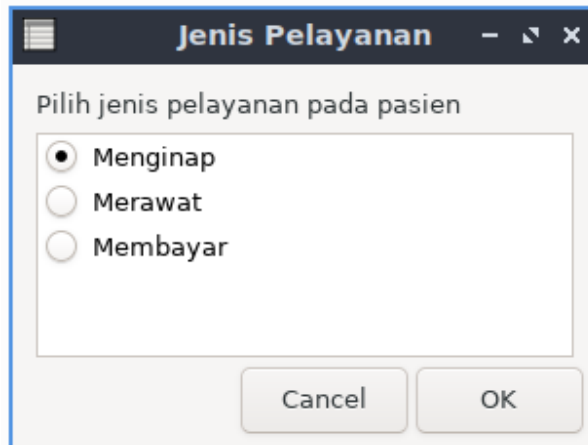
Source code:

```
else zenity --error --title="Gagal Login" --text "Masukkan akun yang benar!" --  
width=200 --height=70  
exit 0;
```

Apabila proses memuat telah selesai(selama 5 detik) maka selanjutnya program mengeksekusi perulangan sebanyak tak hingga yang di dalamnya memanggil fungsi home. Fungsi home ini terdapat subprogram yang menjalankan tampilan menu utama/fitur dari aplikasi ini. Fitur tersebut antara lain Pelayanan, Rekam Medik, Database, Edit Database, History, dan Logout untuk keluar dari aplikasi. Ketika user memilih menu Pelayanan seperti di bawah ini



Maka kemudian akan ditampilkan pilihan pelayanan yang dimaksudkan user, antara lain pelayanan untuk menginap, merawat, atau membayar. Selanjutnya memanggil fungsi main, apabila user memilih pilihan Pelayanan maka akan muncul tampilan seperti di bawah ini



Jika user memilih pilihan menginap selanjutnya akan ditampilkan borang biodata pasien, seperti gambar di bawah ini

Source code pada langkah ini yaitu seperti di bawah ini

Source code:

```
if [[ $fitur == "Pelayanan" ]]; then
    pasien=$(zenity --list --title="Jenis Pelayanan" --text="Pilih jenis pelayanan pada
pasien" \
    --radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE
    "Menginap" FALSE "Merawat" FALSE "Membayar")
```

```

if [ $pasien == "Menginap" ]; then
datap pasien=$(zenity --forms --title="Data Pasien" --text="Masukkan data pasien"
\
--add-entry="Nama Pasien" \
--add-combo="Jenis Kelamin" --combo-values="Laki-Laki|Perempuan"\
--add-entry="Usia" \
--add-entry="Alamat" \
--add-entry="Keluhan" \
--add-calendar="Tanggal")
IFS="|" read -r nama[i] jenis[i] usia[i] alamat[i] keluhan[i] d <<< "$datap pasien" ;
tanggal[i]="20${d:6:2}-${d:3:2}-${d:0:2}"
no[i]=$i;
dok[i]=0
per[i]=0
status[i]="Dirawat"
x[i]=0
if [[ ${nama[i]} != "" ]]; then
sis;

```

Inputan user ini disimpan pada variabel array dengan index i, yang mana nilai i ini akan selalu bertambah dengan ketentuan kolom nama tidak kosong dan user telah memilih kamar pasien. Seperti pada source code di bawah ini

#### Source code:

```

if [[ ${nama[i]} != "" && ${kamarr[i]} != "" ]]; then
let i=$i+1;
fi

```

Nilai i ini juga digunakan sebagai identitas nomor pasien.

Setelah itu seperti halnya yang tertulis pada baris terakhir source code tampilan ini, selanjutnya program memanggil fungsi sis. Fungsi sis ini berisi program untuk menampilkan data ketersediaan kamar rawat inap dari kelas 1, kelas 2, dan kelas 3. Kelas 1 merupakan kamar dengan fasilitas lebih sehingga harga yang ditawarkan lebih mahal, harga semakin menurun hingga kelas 3. Tampilan program sebagai berikut

Kelas 1	Kelas 2	Kelas 3
Bougenville=20 tersisa	Sedudo=30 tersisa	Semeru=50 tersisa
Alamanda=20 tersisa	Roro Kuning=30 tersisa	Wilis=50 tersisa
Anggrek=20 tersisa	Singokromo=30 tersisa	Bromo=50 tersisa

Source code:

```
zenity --list --height=260 --width=483 --title="Data Ketersediaan Kamar" --
text="Pastikan terdapat kamar kosong untuk pasien" \
--column="Kelas 1" --column="Kelas 2" --column="Kelas 3"
"Bougenville=$bou tersisa" "Sedudo=$se tersisa" "Semeru=$seme tersisa"
"Alamanda=$sala tersisa" "Roro Kuning=$ro tersisa" "Wilis=$wili tersisa"
"Anggrek=$sang tersisa" "Singokromo=$si tersisa" "Bromo=$brom tersisa"
```

Banyaknya kamar pada tampilan di atas yaitu disimpan dalam variabel yang nilainya telah dideklarasikan pada awal program berjalan. Selanjutnya apabila user memilih tombol OK maka fungsi kelas akan dipanggil yang akan menampilkan pilihan kelas yang tersedia seperti pada data diatas, tampilannya sebagai berikut



Source code:

```
kelas() {
kelas=$(zenity --list --title="Pilih Kelas Kamar" --text="<b>Tarif kamar :</b>
\nKelas 1 = Rp 3.000.000/hari\nKelas 2 = Rp 1.500.000/hari\nKelas 3 = Rp
500.000/hari" \
--radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE "Kelas 1"
FALSE "Kelas 2" FALSE "Kelas 3")
if [ $? -eq 1 ]; then
sis;
else kamar;
fi
}
```

Pada simulasi program ini, pasien nomor urut 1 memilih kelas 1. Selanjutnya akan memanggil fungsi kamar dan memunculkan pilihan kamar dari kelas 1

Source code:

```
if [[ $kelas == "Kelas 1" ]]; then
kmr=$(zenity --list --title="Kamar Kelas 1 Tersedia" --text="Tarif kamar kelas 1
Rp 3000.000/hari" \
--radiolist --hide-header --column "Pilih" --column "Pelayanan" TRUE
"Bougenville" FALSE "Alamanda" FALSE "Anggrek")
let kali[i]=3000000
IFS="|" read -r kamarr[i] <<< "$kmr" ;
```



```

if [[ ${kamarr[i]} == "Bougenville" && $bou > 0 ]];then
kamar[i]=${kamarr[i]}$bou
let bou=$bou-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Alamanda" && $sala > 0 ]];then
kamar[i]=${kamarr[i]}$sala
let ala=$sala-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70
elif [[ ${kamarr[i]} == "Anggrek" && $sang > 0 ]];then
kamar[i]=${kamarr[i]}$sang
let ang=$sang-1
zenity --info --title="Sukses" --text "Data pasien telah tersimpan" --width=200 --
height=70

```

Output dari source code tersebut seperti gambar di bawah ini



Dalam simulasi ini user memilih kamar Bougenville, ketika tombol OK dipilih maka variabel bou akan berkurang satu, sehingga kini kamar bougenville tersisa 19.

**Data Ketersediaan Kamar**


Pastikan terdapat kamar kosong untuk pasien

Kelas 1	Kelas 2	Kelas 3
Bougenville=19 tersisa	Sedudo=30 tersisa	Semeru=50 tersisa
Anggrek=20 tersisa	Singokromo=30 tersisa	Bromo=50 tersisa
Alamanda=20 tersisa	Roro Kuning=30 tersisa	Wilis=50 tersisa

Cancel OK

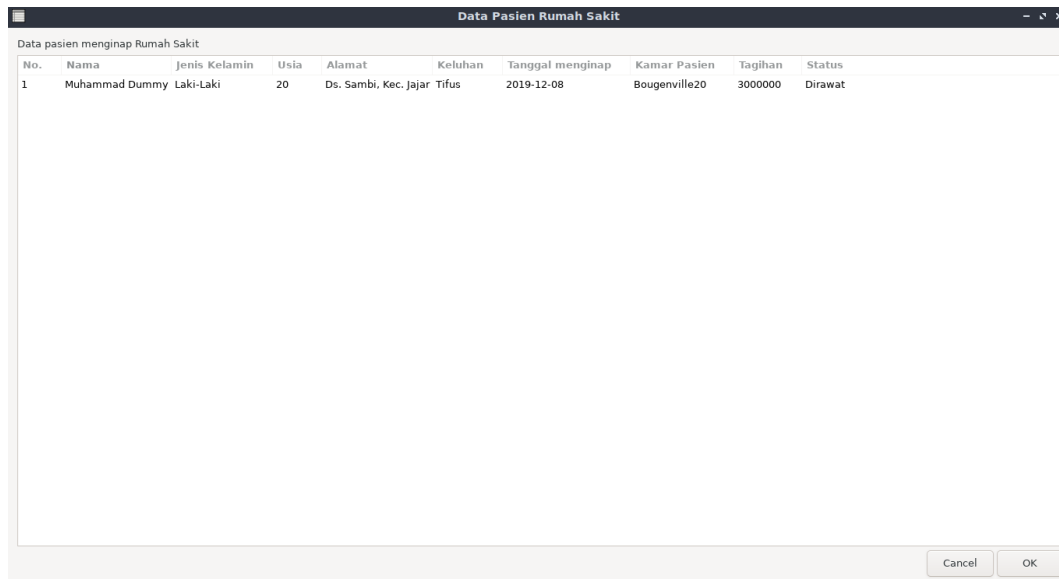
Setelah data-data telah berhasil diisi maka akan muncul notifikasi sukses dan data tersimpan pada database.

**Sukses**

 Data pasien telah tersimpan

OK

Tampilan database pasien seperti di bawah ini

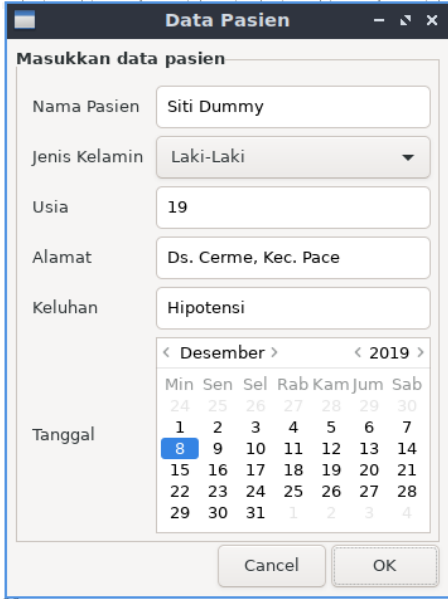


No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal menginap	Kamar Pasien	Tagihan	Status
1	Muhammad Dummy	Laki-Laki	20	Ds. Sambi, Kec. Jajar	Tifus	2019-12-08	Bougenville20	3000000	Dirawat

Dalam tabel database ini ditampilkan nomor pasien, nama, jenis kelamin, usia, alamat, keluhan, tanggal mulai rawat inap, kamar, tagihan, dan status pasien(Masih dirawat/sudah keluar). Tagihan yang ditampilkan ini menghitung otomatis biaya perawatan pasien selama di rumah sakit antara lain akumulasi biaya kamar per hari yang dihitung otomatis *realtime* dari tanggal awal masuk hingga tanggal hari ini pasien dirawat dan ditambah biaya perawatan oleh dokter ataupun perawat. Akan tetapi database ini tidak bersifat permanen karena data-data tersebut disimpan dalam variabel array, apabila menggunakan database yang disimpan dalam txt, data-data pasien tersebut tidak dapat diubah(hanya dapat ditambahkan) sehingga kurang cocok digunakan. Penggunaan database permanen dalam bentuk data txt pada aplikasi ini yaitu pada menu History, pada menu history yaitu menampilkan data pasien yang telah keluar dan membayar tagihan rumah sakit. Untuk penjelasan lebih lanjut akan dibahas selanjutnya.

Pada simulasi ini terdapat pasien kedua, dengan langkah seperti di bawah ini

Mengisikan biodata



The 'Data Pasien' form contains the following fields and values:

- Nama Pasien: Siti Dummy
- Jenis Kelamin: Laki-Laki
- Usia: 19
- Alamat: Ds. Cerme, Kec. Pace
- Keluhan: Hipotensi
- Tanggal: A calendar for December 2019 with the 8th selected.

Buttons: Cancel, OK

Memilih kelas



The 'Pilih Kelas Kamar' form displays the following information:

**Tarif kamar :**  
Kelas 1 = Rp 3.000.000/hari  
Kelas 2 = Rp 1.500.000/hari  
Kelas 3 = Rp 500.000/hari

Selection options:

- ☐ Kelas 1
- ☒ Kelas 2
- ☐ Kelas 3

Buttons: Back, OK

Memilih kamar



Kamar K...Tersedia

Tarif kamar kelas 2 Rp 1.500.000/hari

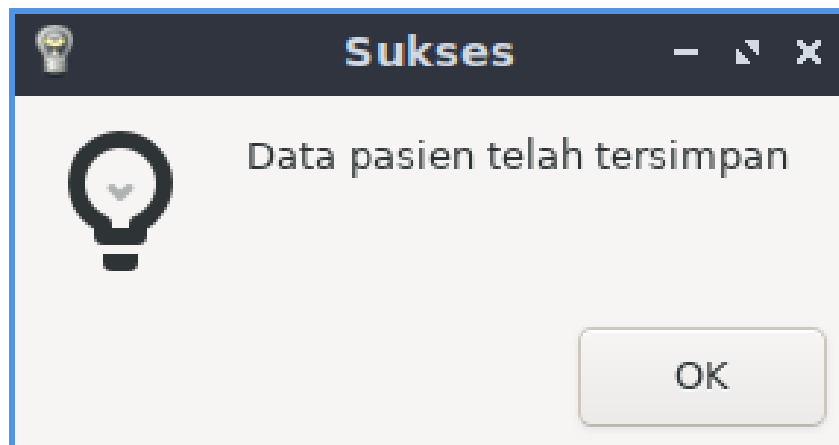
☐ Sedudo

☒ Roro Kuning

☐ Singokromo

Back OK

Data Tersimpan



Sukses

Data pasien telah tersimpan

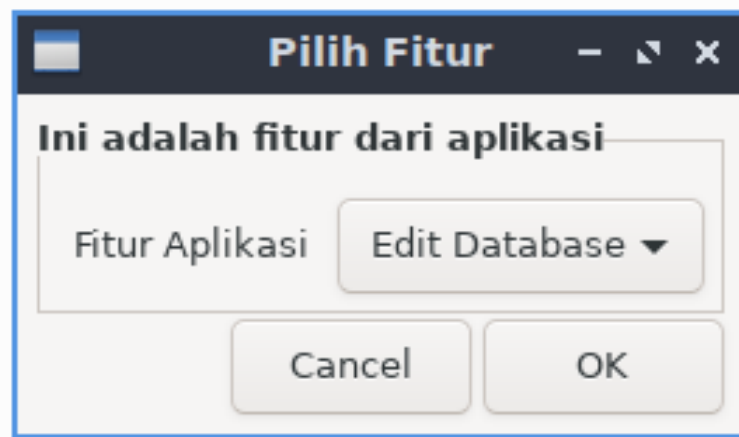
OK

Tampilan database akan bertambah, seperti gambar di bawah ini



No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal menginap	Kamar Pasien	Tagihan	Status
1	Muhammad Dummy	Laki-Laki	20	Ds. Sambu, Kec. Jajar	Tifus	2019-12-08	Bougenville20	3000000	Dirawat
2	Siti Dummy	Laki-Laki	19	Ds. Cerme, Kec. Pace	Hipotensi	2019-12-08	Roro Kuning30	1500000	Dirawat

Pada database diatas terlihat bahwa pasien nomor 2 terdapat kesalahan pada pemilihan jenis kelamin saat menginputkan data, untuk melakukan perubahan maka dipilih menu Edit Database



Menu edit database menggunakan source code seperti di bawah ini

Source code:

```
datapasi en=$ (zenity --forms --title="Data Pasien" --text="Masukkan data pasien"
\
--add-entry="Nomor Urut Pasien" \
--add-entry="Nama Pasien" \
--add-combo="Jenis Kelamin" --combo-values="Laki-Laki|Perempuan"\
--add-entry="Usia" \
--add-entry="Alamat" \
--add-entry="Keluhan" \
--add-calendar="Tanggal")
IFS="|" read -r noo1 nama1 jenis1 usia1 alamat1 keluhan1 d <<< "$datapasi en" ;
tanggal="20${d:6:2}-${d:3:2}-${d:0:2}"
if [[ $noo1 == ${no[$noo1]} && $noo1 != "" ]];then
no[$noo1]=$noo1 nama[$noo1]=${nama1} jenis[$noo1]=${jenis1}
usia[$noo1]=${usia1} alamat[$noo1]=${alamat1} keluhan[$noo1]=${keluhan1}
tanggal[$noo1]=${tanggal}
zenity --info --title="Sukses" --text "Data pasien telah diperbaiki" --width=200 --
height=70
```

Dengan source code diatas akan menghasilkan tampilan sebagai berikut

**Data Pasien**

Masukkan data pasien

Nomor Urut Pasien: 2

Nama Pasien: Siti Dummy

Jenis Kelamin: Perempuan

Usia: 19

Alamat: Ds. Cerme, Kec. Pace

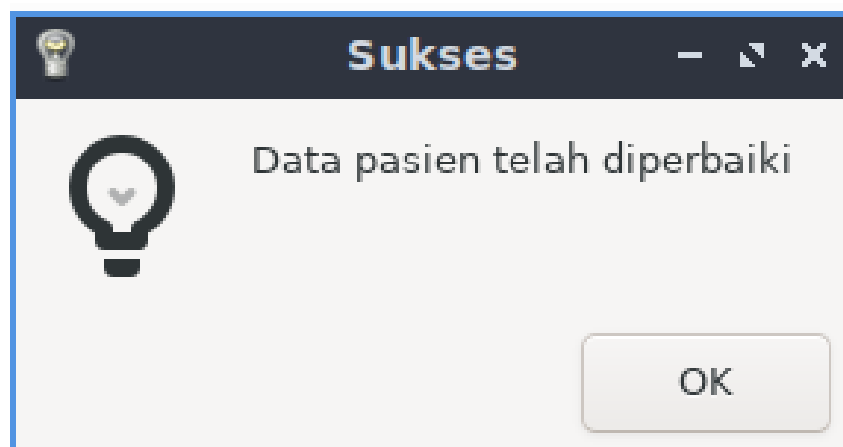
Keluhan: Hipotensi

Tanggal: < Desember > < 2019 >

Min	Sen	Sel	Rab	Kam	Jum	Sab
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

Buttons: Cancel, OK

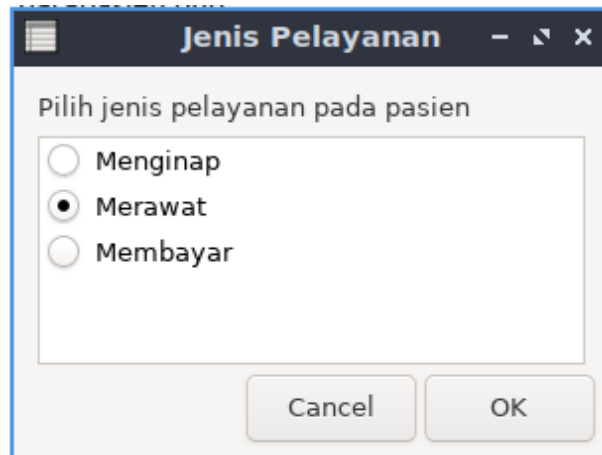
Pada borang diatas telah dipastikan data yang diinputkan adalah benar, selanjutnya data akan diperbaharui dan muncul tampilan berhasil memperbaharui data, seperti gambar di bawah ini



Tampilan pada menu Database setelah dilakukan perubahan data

Data Pasien Rumah Sakit									
Data pasien menginap Rumah Sakit									
No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal menginap	Kamar Pasien	Tagihan	Status
1	Muhammad Dummy	Laki-Laki	20	Ds. Sambi, Kec. Jajar	Tifus	2019-12-08	Bougenville20	3000000	Dirawat
2	Siti Dummy	Perempuan	19	Ds. Cerme, Kec. Pace	Hipotensi	2019-12-08	Roro Kuning30	1500000	Dirawat

Simulasi selanjutnya yaitu pilihan Merawat pada menu Pelayanan



Pilihan merawat ini menggunakan source code seperti di bawah ini

Source code:

```
rekampasien=$(zenity --forms --title="Data Pasien" --text="Masukkan data
pasien" \
--add-entry="Nomor Urut Pasien" \
--add-calendar="Tanggal Periksa" \
--add-entry="Dokter"\
--add-entry="Perawat"\
--add-entry="Tekanan Darah"\
--add-entry="Suhu Badan")
IFS="|" read -r noo1 d dokter1 perawat1 tensi1 suhu1 <<< "$rekampasien" ;
tglp1="20${d:6:2}-${d:3:2}-${d:0:2}"
if [ $noo1 == ${no[$noo1]} ]; then
x[$noo1]=$((x[$noo1]+1))
if [[ $dokter1 != "" ]]; then
noo[$noo1]=$noo1 dokter[$noo1,$x[$noo1]]=$dokter1
perawat[$noo1,$x[$noo1]]=$perawat1 tglp[$noo1,$x[$noo1]]=$tglp1
tensi[$noo1,$x[$noo1]]=$tensi1 suhu[$noo1,$x[$noo1]]=$suhu1
dok[$noo1]=$((dok[$noo1]+500000))
periksa[$noo1,$x[$noo1]]="Ke ${x[$noo1]}"
zenity --info --title="Sukses" --text "Rekam medik pasien oleh dokter telah
tersimpan" --width=200 --height=70
elif [[ $perawat1 != "" ]]; then
```



```

noo[$noo1]=$noo1 dokter[$noo1,$x[$noo1]]=$dokter1
perawat[$noo1,$x[$noo1]]=$perawat1 tglp[$noo1,$x[$noo1]]=$tglp1
tensi[$noo1,$x[$noo1]]=$tensi1 suhu[$noo1,$x[$noo1]]=$suhu1
per[$noo1]=$((per[$noo1]+250000))
periksa[$noo1,$x[$noo1]]="Ke $x[$noo1]"
zenity --info --title="Sukses" --text "Rekam medik pasien oleh perawat telah
tersimpan" --width=200 --height=70

```

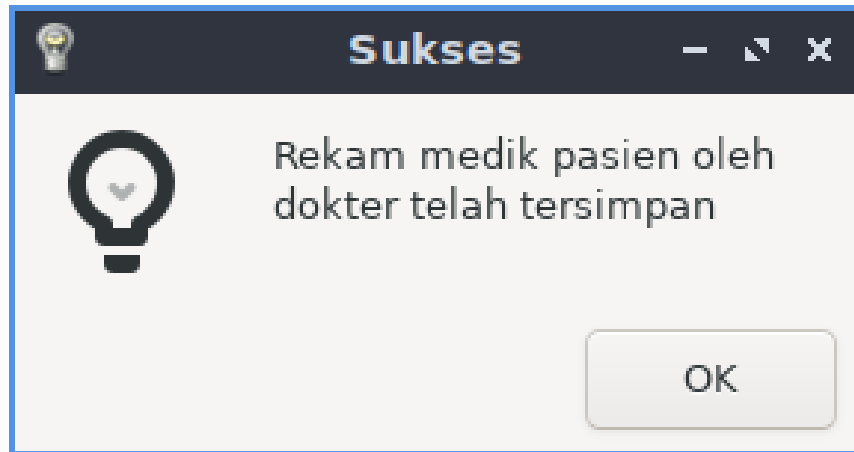
Tampilan dari source code diatas adalah seperti pada gambar berikut

The screenshot shows a window titled "Data Pasien" with a form titled "Masukkan data pasien". The form contains the following fields and values:

- Nomor Urut Pasien: 1
- Tanggal Periksa: A calendar for December 2019 with the 8th selected.
- Dokter: dr. Darmawan
- Perawat: (empty field)
- Tekanan Darah: 110
- Suhu Badan: 40

At the bottom of the form are "Cancel" and "OK" buttons.

Pada nomor urut pasien diberikan input angka 1 sehingga ini adalah pasien pertama yaitu Muhammad Dummy. Apabila user memilih tombol OK maka akan muncul tampilan



Pada source code diatas tertulis apabila kolom dokter diberikan isian maka variabel array dokter[i] akan bertambah 500000 sedangkan apabila kolom perawat diberikan isian maka variabel array per[i] akan bertambah 250000. Sehingga total tagihan pada database akan bertambah.

Tampilan database

No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal menginap	Kamar Pasien	Tagihan	Status
1	Muhammad Dummy	Laki-Laki	20	Ds. Sambi, Kec. Jajar	Tifus	2019-12-08	Bougenville20	3500000	Dirawat
2	Siti Dummy	Laki-Laki	19	Ds. Cerme, Kec. Pace	Hipotensi	2019-12-08	Roro Kuning30	1500000	Dirawat

Tagihan pasien nomor urut 1 berubah dari 3000000 menjadi 3500000.

Data-data pemeriksaan pasien yang dirawat ditampilkan dalam menu Rekam Medik, source code menu rekam medik adalah sebagai berikut

Source code:

```
for o in $(seq 1 $i)
do
for v in $(seq 1 ${x[o]})
do
if [ ${noo[o]} != "" ]; then
if [ ${noo[o]} == ${no[o]} ];then
echo ${noo[o]}
echo ${nama[o]}
echo ${tglp[$o,$v]}
```

```

echo ${dokter[$o,$v]}
echo ${perawat[$o,$v]}
echo ${tensi[$o,$v]}
echo ${suhu[$o,$v]}
echo ${kamar[o]}
echo ${periksa[$o,$v]}
fi
fi
done

done | zenity --list --height=720 --width=1366 --title="Data Pasien Rumah Sakit"
--text="Data pasien menginap Rumah Sakit"\
--column="No." --column="Nama" --column="Tanggal Periksa" --
column="Dokter" --column="Perawat" --column="Tekanan Darah" --
column="Suhu Badan" --column="Kamar Pasien" --column="Periksa"

```

Dengan source code diatas menghasilkan tampilan sebagai berikut



The screenshot shows a terminal window titled "Data Pasien Rumah Sakit". Inside the window, a table is displayed with the following data:

No.	Nama	Tanggal Periksa	Dokter	Perawat	Tekanan Darah	Suhu Badan	Kamar Pasien	Periksa
1	Muhammad Dummy	2019-12-08	dr. Darmawan		110	40	Bougenville20	Ke 1

Simulasi selanjutnya yaitu merawat pasien nomor urut 2

Tampilan borang merawat pasien

**Data Pasien**

Masukkan data pasien

Nomor Urut Pasien: 2

Tanggal Periksa: < Desember > < 2019 >

Min	Sen	Sel	Rab	Kam	Jum	Sab
24	25	26	27	28	29	30
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4

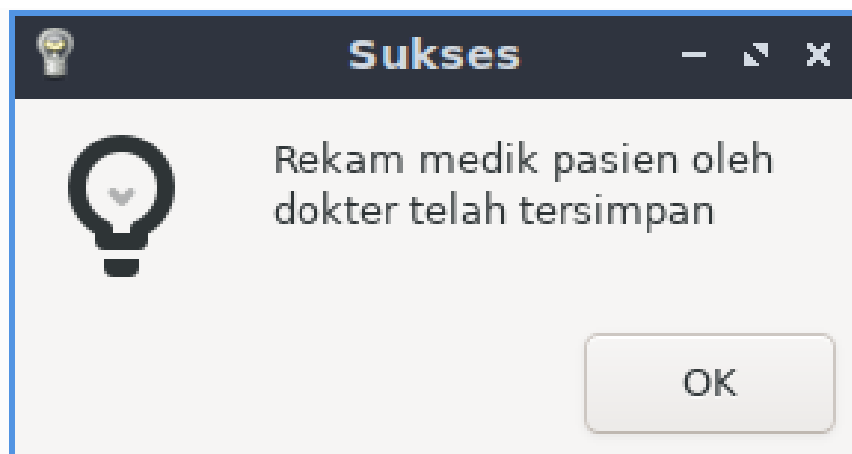
Dokter:

Perawat: Raisa A.Md.Kep.

Tekanan Darah: 90/60 mmHg

Suhu Badan:

Cancel OK



Tampilan database Rekam Medik akan menjadi seperti berikut

Data Pasien Rumah Sakit								
Data pasien menginap Rumah Sakit								
No.	Nama	Tanggal Periksa	Dokter	Perawat	Tekanan Darah	Suhu Badan	Kamar Pasien	Periksa
1	Muhammad Dummy	2019-12-08	dr. Darmawan		110	40	Bougenville20	Ke 1
2	Siti Dummy	2019-12-08		Raisa A.Md.	90/60 mmHg		Roro Kuning30	Ke 1

Selanjutnya yaitu merawat pasien nomor urut 1 kembali

Tampilan borang merawat pasien

Tampilan database Rekam Medik akan menjadi seperti berikut

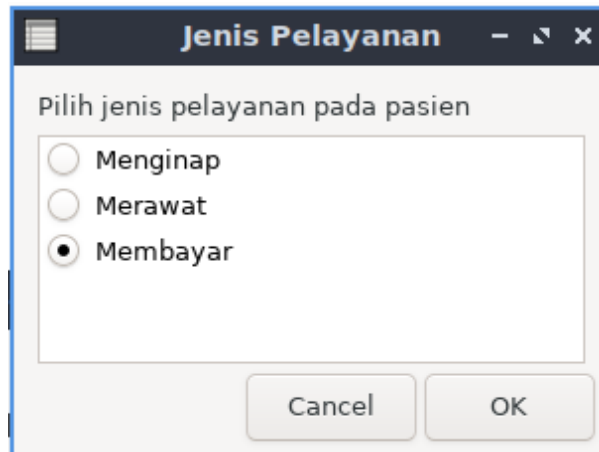
No.	Nama	Tanggal Periksa	Dokter	Perawat	Tekanan Darah	Suhu Badan	Kamar Pasien	Periksa
1	Muhammad Dummy	2019-12-08	dr. Damawan		110	40	Bougenville20	Ke 1
1	Muhammad Dummy	2019-12-09		Isyana A.Md.	115/70 mmHg	38	Bougenville20	Ke 2
2	Siti Dummy	2019-12-08		Raisa A.Md.	90/60 mmHg		Roro Kuning30	Ke 1

Hal ini karena penggunaan variabel array 2 dimensi sehingga data perawatan pertama pada pasien nomor urut 1 tidak hilang.

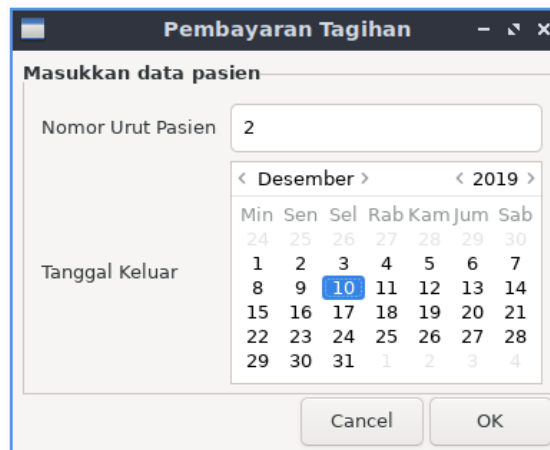
Karena setiap perawatan menambah biaya tagihan, maka tampilan database pasien akan menjadi seperti berikut

No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal menginap	Kamar Pasien	Tagihan	Status
1	Muhammad Dummy	Laki-Laki	20	Ds. Sambu, Kec. Jajar	Tifus	2019-12-08	Bougenville20	3750000	Dirawat
2	Siti Dummy	Perempuan	19	Ds. Cerme, Kec. Pace	Hipotensi	2019-12-08	Roro Kuning30	1750000	Dirawat

Simulasi selanjutnya yaitu pasien nomor urut 2 melakukan pembayaran, langkahnya yaitu user memilih menu Pelayanan kemudian pilihan Membayar



Selanjutnya akan muncul tampilan seperti pada gambar di bawah ini



User memasukkan nomor urut pasien serta memilih tanggal keluar pasien yang bersangkutan, pada simulasi ini yaitu dipilih tanggal 10 Desember 2019. Pada tampilan ini yaitu dibuat dengan source code seperti di bawah ini

Source code:

```
biayapasien=$(zenity --forms --title="Pembayaran Tagihan" --text="Masukkan
data pasien" \
--add-entry="Nomor Urut Pasien" \
--add-calendar="Tanggal Keluar" )
IFS="|" read -r noo1 d <<< "$biayapasien" ;
tglk[$noo1]="20${d:6:2}-${d:3:2}-${d:0:2}"
if [ $noo1 == ${no[$noo1]} ]; then
```

```

keluar=$(date -d "${tglk[$$noo1]}" "+%s")
masuk=$(date -d "${tanggal[$$noo1]}" "+%s")
let hari=$(( ( $keluar/86400 ) - ( $masuk/86400 ) )+1)
let kamar=($ {kali[$$noo1]}*$hari)
let penanganan=($ {dok[$$noo1]}+$ {per[$$noo1]})
let biaya=($kamar+$penanganan)
zenity --info --title="Total Bayar" --text "<b>Rincian biaya :</b>\nKamar =
$skamar \nPenanganan = $penanganan\n<b>Total yang harus dibayar =
$biaya</b>" --width=500 --height=70
no[$$noo1]=" "
status[$$noo1]="Keluar"
if [[ $ {kamarr[$$noo1]} == "Bougenville" ]];then
let bou=$bou+1;
elif [[ $ {kamarr[$$noo1]} == "Alamanda" ]];then
let ala=$ala+1;
elif [[ $ {kamarr[$$noo1]} == "Anggrek" ]];then
let ang=$ang+1;
elif [[ $ {kamarr[$$noo1]} == "Sedudo" ]];then
let se=$se+1;
elif [[ $ {kamarr[$$noo1]} == "Roro Kuning" ]];then
let ro=$ro+1;
elif [[ $ {kamarr[$$noo1]} == "Singokromo" ]];then
let si=$si+1;
elif [[ $ {kamarr[$$noo1]} == "Semeru" ]];then
let seme=$seme+1;
elif [[ $ {kamarr[$$noo1]} == "Wilis" ]];then
let wili=$wili+1;
elif [[ $ {kamarr[$$noo1]} == "Bromo" ]];then
let brom=$brom+1;
fi
echo "$ {no[$$noo1]}
$ {nama[$$noo1]}|$ {jenis[$$noo1]}|$ {usia[$$noo1]}|$ {alamat[$$noo1]}|$ {keluhan[$$noo1]}|$ {status[$$noo1]}"

```

```

nool}}|${tanggal[$nool]}|${tglk[$nool]}|$hari|$biaya" >>
/home/hdr/snap/dataRS
else
zenity --error --title="Galat" --text "Pasien tidak terdaftar" --width=200 --
height=70
fi

```

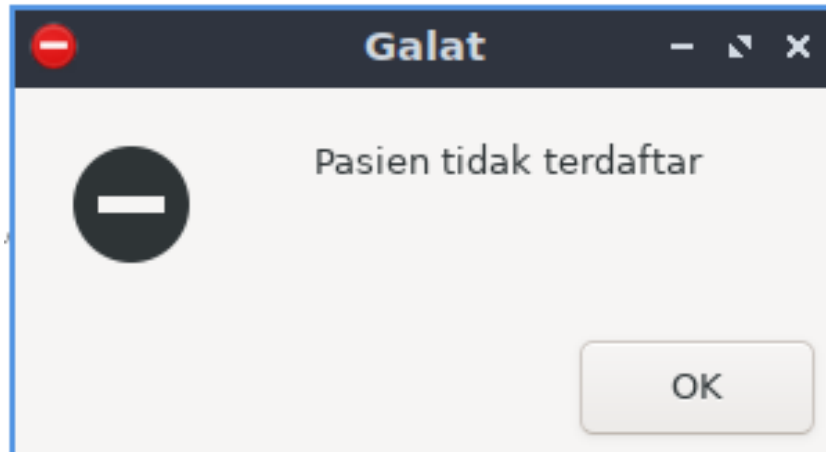
Pada source code diatas dapat dianalisis apabila inputan nomor pasien benar maka akan dilakukan penghitungan banyaknya hari saat pasien dirawat kemudian dikalikan dengan tarif kamar tiap harinya serta dijumlahkan dengan biaya perawatan oleh dokter atau perawat pada pasien yang bersangkutan. Selain itu akan dilakukan penjumlahan variabel kamar, sehingga jumlah kamar yang tersedia akan bertambah karena pasien yang bersangkutan telah keluar. Dan data pasien tersebut akan di ekspor pada file txt yang dijadikan database, file txt ini akan ditampilkan pada menu history.

Apabila user memilih OK maka akan muncul notifikasi biaya yang harus dibayar oleh pasien, seperti pada gambar di bawah ini



Apabila user salah memasukkan nomor maka akan muncul peringatan seperti di bawah ini

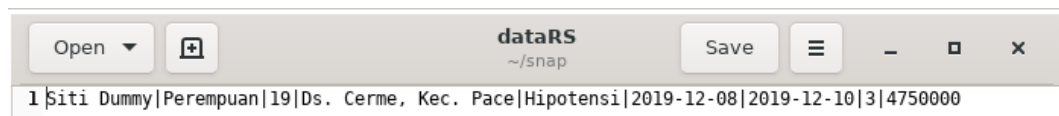




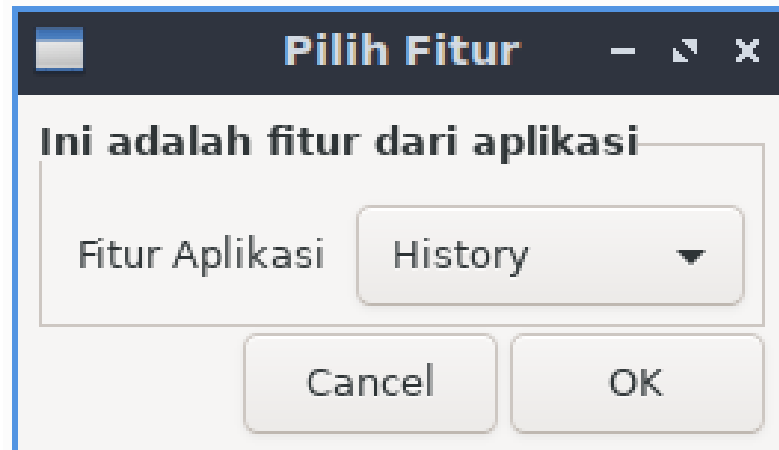
Tampilan database akan berubah menjadi seperti gambar di bawah ini, biaya tagihan pada database tidak sama dengan biaya tagihan yang dibayar karena pasien keluar bukan pada tanggal yang sama dengan system, database menampilkan tagihan sesuai dengan tanggal hari ini.

No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal menginap	Kamar Pasien	Tagihan	Status
1	Muhammad Dummy	Laki-Laki	20	Ds. Sambu, Kec. Jajar	Tifus	2019-12-08	Bougenville20	3750000	Dirawat
	Siti Dummy	Perempuan	19	Ds. Cerme, Kec. Pace	Hipotensi	2019-12-08	Roro Kuning30	1750000	Keluar

Terdapat output txt yang isinya sebagai berikut



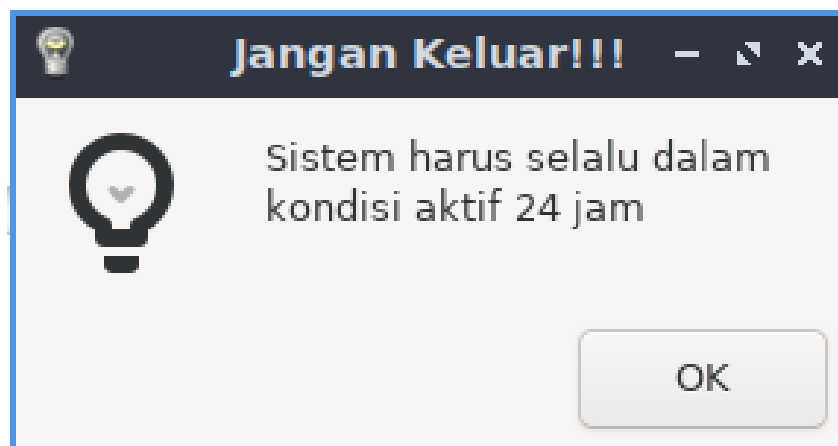
Selanjutnya yaitu memilih menu History untuk melihat apakah data sudah masuk database permanen



Menu History(database permanen) akan tampil seperti berikut

History Rawat Inap								
Data pasien yang pernah menginap di Rumah Sakit								
No.	Nama	Jenis Kelamin	Usia	Alamat	Keluhan	Tanggal Masuk	Tanggal Keluar	Lama Menginap
1	Siti Dummy	Perempuan	19	Ds. Cerme, Kec. Pace	Hipotensi	2019-12-08	2019-12-10	3
								Biaya
								4750000

Simulasi selanjutnya yaitu keluar dari aplikasi. Langkah untuk keluar aplikasi bukan dengan memilih tombol cancel pada menu utama, apabila pada menu utama dipilih cancel maka akan muncul peringatan seperti gambar di bawah ini



Hal ini karena source code ini pada menu utama

Source code:

```
if [ $? -eq 1 ]; then
```

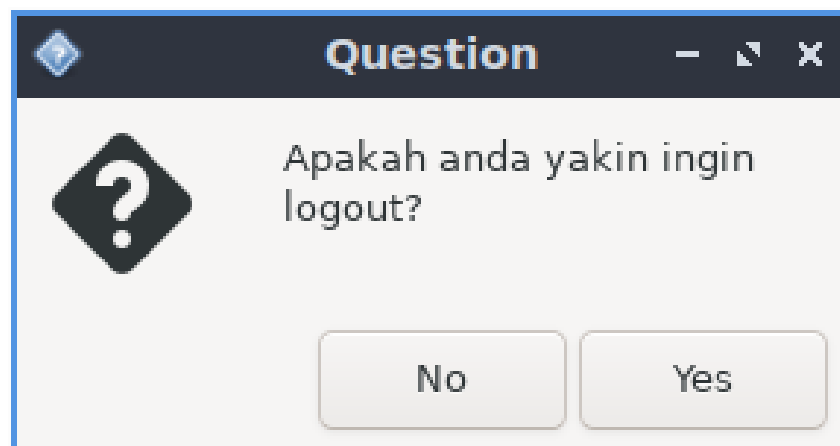
```
zenity --info --title="Jangan Keluar!!!" --text "Sistem harus selalu dalam kondisi aktif 24 jam" --width=200 --height=70
```

fi

Langkah untuk keluar dari aplikasi adalah dengan memilih menu log out pada menu utama seperti di bawah ini

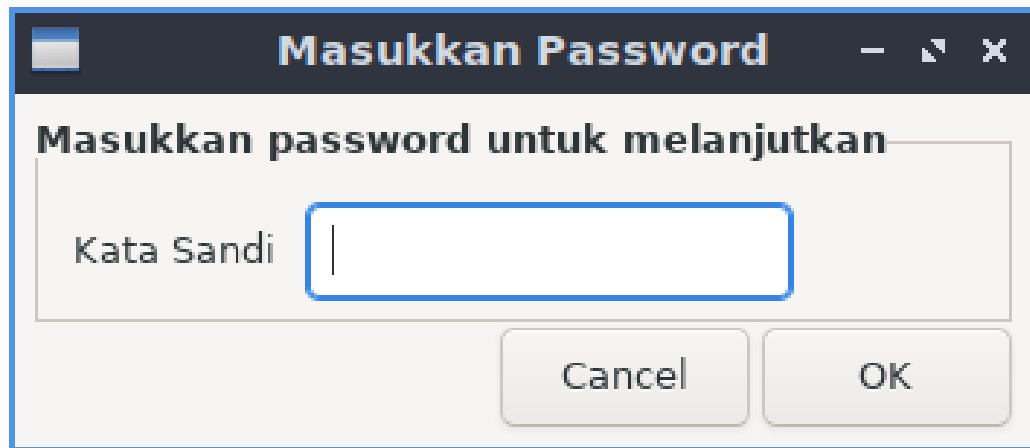


Apabila user memilih tombol OK maka akan muncul tampilan seperti gambar di bawah ini



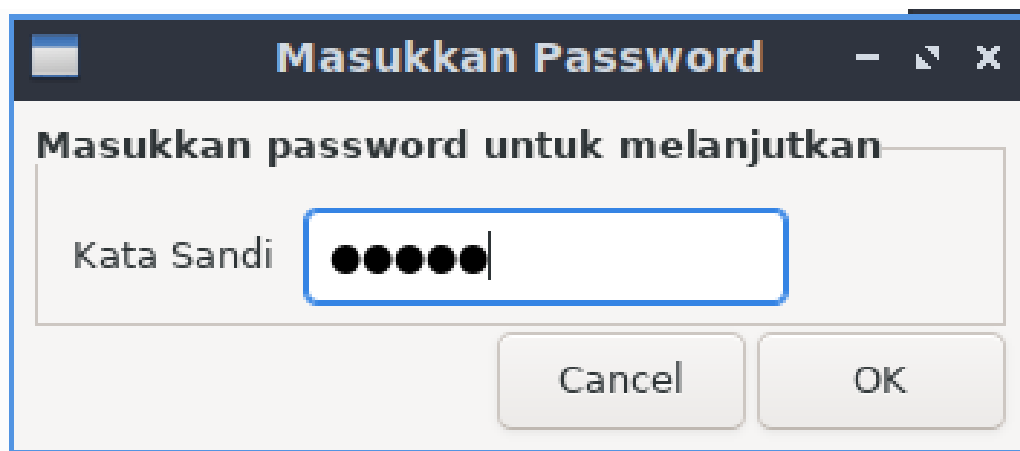
Apabila dipilih yes maka akan muncul kolom inputan kata sandi seperti pada gambar di bawah ini

Tampilan inputan kata sandi

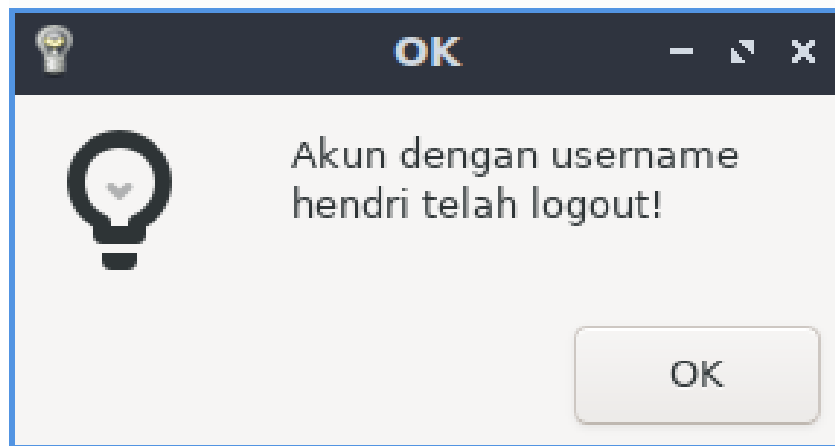


Kata sandi yang dimasukkan adalah harus sama dengan kata sandi yang digunakan untuk login dalam artian hanya satu kata sandi yang valid antara “waras” atau “sehat” tergantung inputan kata sandi pilihan yang dimasukkan user.

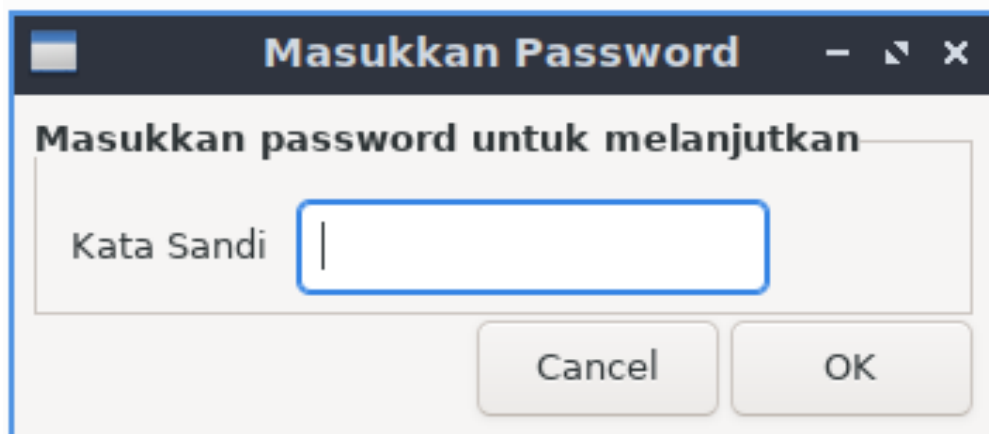
Tampilan user memasukkan kata sandi adalah seperti pada gambar di bawah ini



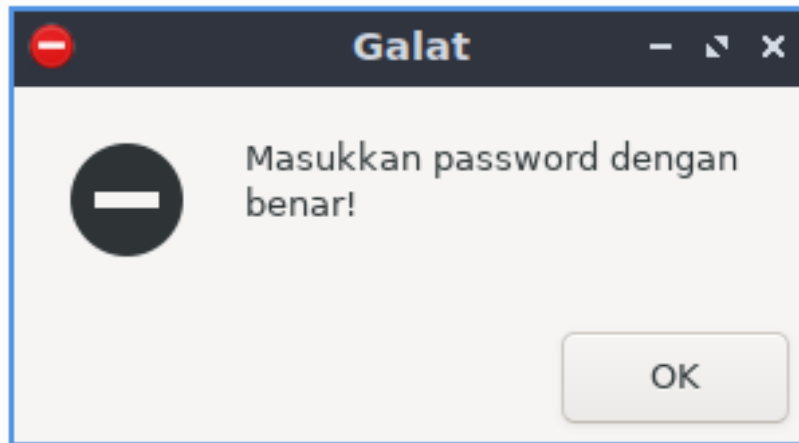
Jika inputan benar maka program akan muncul notifikasi bahwa logout berhasil, seperti pada gambar di bawah ini kemudian apabila dipilih tombol OK maka program akan keluar



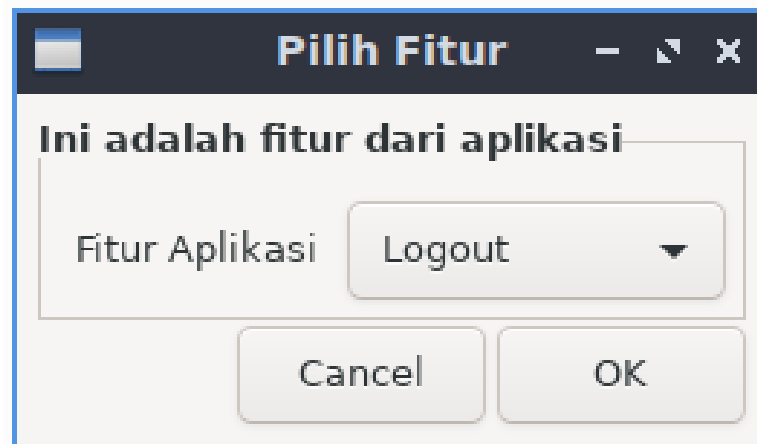
Apabila inputan kata sandi dikosongkan atau inputan salah, misalnya seperti pada gambar di bawah ini



Ketika program dijalankan maka akan muncul peringatan seperti pada gambar di bawah ini, karena kata sandi yang diinputkan kosong



Kemudian program akan kembali ke menu utama



Source code yang digunakan pada tampilan logout adalah sebagai berikut

Source code:

```
zenity --question --text="Apakah anda yakin ingin logout?" --default-cancel --
width=200 --height=70
if [ $? -eq 0 ];then
pwod2=$(zenity --forms --width=350 --title="Masukkan Password" --
text="Masukkan password untuk melanjutkan" \
--add-password="Kata Sandi")
if [[ $pwod2 == $pwod ]];then
zenity --info --title="OK" --text "Akun dengan username $user telah logout!" --
width=200 --height=70
```

```
exit 0;  
else zenity --error --title="Galat" --text "Masukkan password dengan benar!" --  
width=200 --height=70
```

## **BAB IV**

### **PENUTUP**

#### 4.1 Kesimpulan

Dari hasil *final project* yang telah dikerjakan dapat disimpulkan bahwa aplikasi sederhana Pendataan Pasien Rawat Inap Rumah Sakit dapat dibuat menggunakan pemrograman *shell* dengan tampilan yang tidak kalah menarik dari pemrograman lain. Pemrograman *shell* dapat digunakan untuk membuat GUI dengan zenity, yad, dan lain-lain, akan tetapi yang otomatis terinstal pada system linux hanyalah zenity, sehingga harus mengunduh apabila ingin menggunakan yad. Selain itu, pada pemrograman *shell* terdapat beberapa sintak yang tidak kalah lengkap dengan pemrograman lain serta kemudahan dalam membuat database yang mana sangat membantu dalam penyimpanan data.



## DAFTAR PUSTAKA

- Akuwan, S. 2019. “Pengkondisian”. Dalam *Modul Praktikum Shell Programming Semester III*. Surabaya.
- Akuwan, S. 2019. “Perulangan”. Dalam *Modul Praktikum Shell Programming Semester III*. Surabaya.
- Akuwan, S. 2019. “Array”. Dalam *Modul Praktikum Shell Programming Semester III*. Surabaya.
- Akuwan, S. 2019. “Subrutin atau Fungsi”. Dalam *Modul Praktikum Shell Programming Semester III*. Surabaya.
- Gnome.org. *Zenity Manual*[Internet].*Gnome Library*. Diakses 1 Desember 2019. Tersedia dari <https://help.gnome.org/users/zenity>