BAB III

METODE PENELITIAN DAN PERANCANGAN SISTEM

3.1. Metode Penelitian

Dalam perancangan sistem ini, metodologi penelitian yang penulis pakai adalah sebagai berikut :

a) Analisis

Penulis melakukan analisis permasalahan yang terjadi dan melihat kebutuhan dari sistem yang dibuat untuk menyelesaikan permasalahan. Dimulai dengan melakukan analisis model antrian dan membuat blok diagram.

b) Studi literatur

Penulis mencari literatur untuk mendapatkan informasi-informasi yang berhubungan dengan permasalahan antrian dalam kehidupan sehari-hari melalui buku dan karya tugas akhir sebelumnya.

c) Pengamatan/Observasi

Cara ini dilakukan untuk mengamati dan meneliti cara kerja aplikasi antrian di tempat-tempat pelayanan *customer* guna mengetahui berlangsungnya proses antrian untuk memperoleh

sebuah informasi dalam melakukan pembuatan perangkat lunak sistem antrian yang akan dibuat.

d) Desain

Penulis kemudian merancang hasil analisa dalam bentuk diagram cara kerja aplikasi dan *Use Case Diagram, flowchart* hinggal metode pembuatan aplikasi.

e) Implementasi dan evaluasi

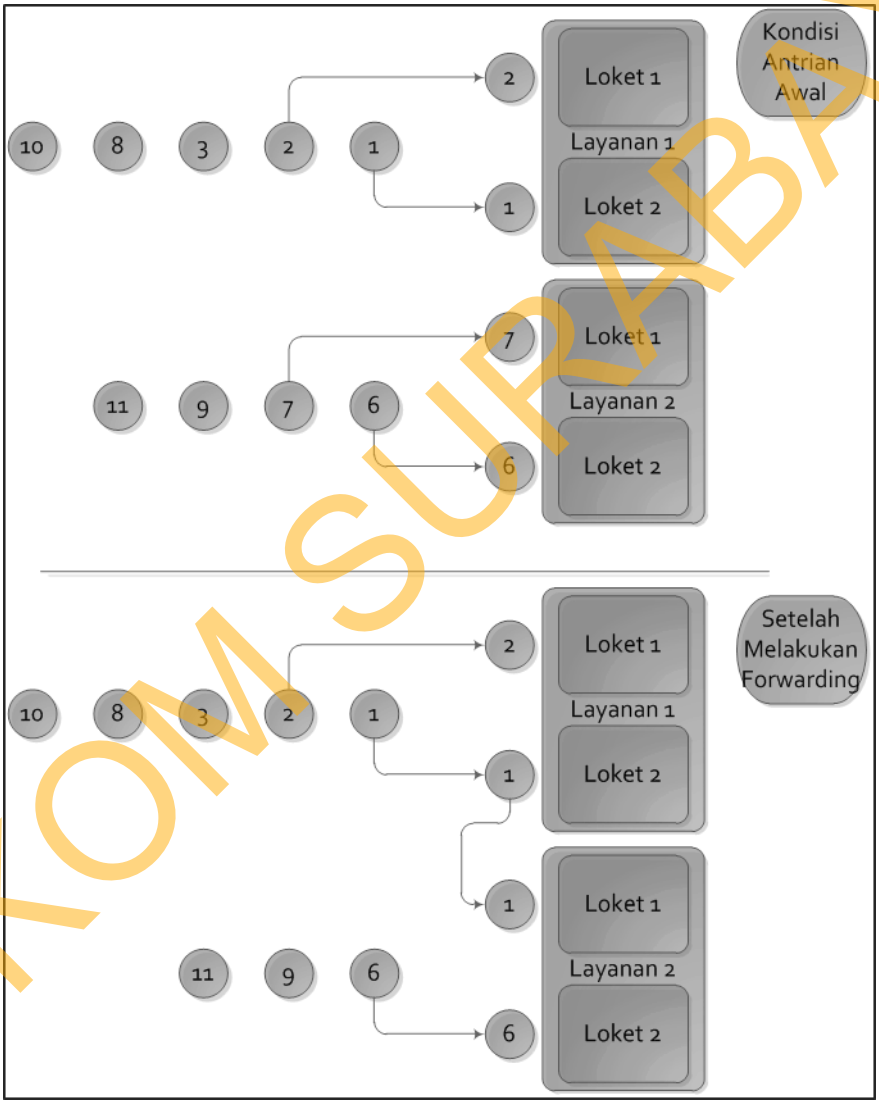
Merupakan tahapan penerapan dan pengujian dari aplikasi sistem antrian yang telah dibuat.

3.2. Analisis Model Antrian

Dalam sistem antrian kali ini akan digunakan model antrian *multiple channel queue* dengan *forwarding*. Dimana aplikasi mampu melakukan *forwarding* nomor antrian ke jalur antrian pada layanan yang berbeda. Misalnya seseorang telah antri dan mendapatkan pelayanan di *Customer Service*, kemudian dia akan diforward ke kasir untuk melakukan pembayaran tanpa perlu mengambil nomor antrian lagi.

Nomor yang diforward tetap akan mendapatkan prioritas sesuai dengan nomor antrian yang telah diambilnya seperti pada gambar 3.1. Nomor antrian 1 sudah dilayani pada Loket 2 di jenis

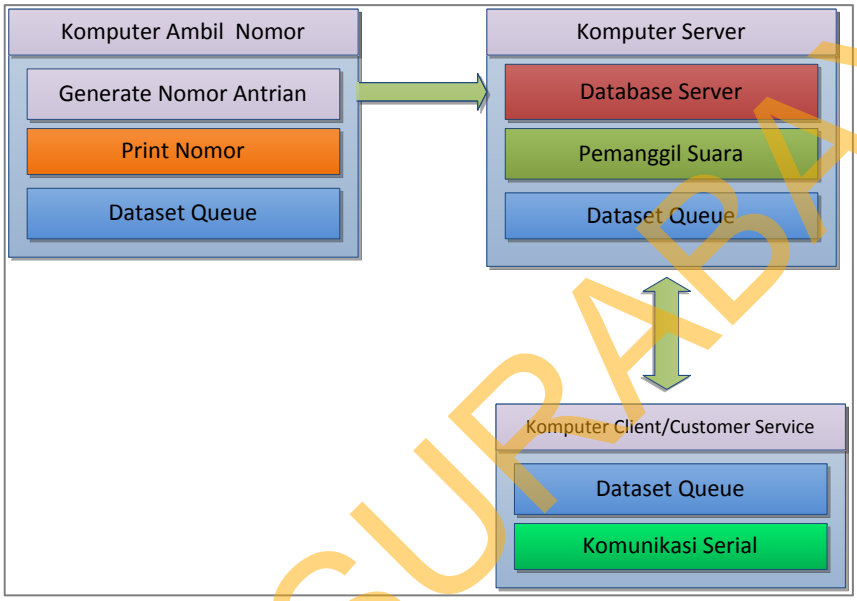
layanan 1. Setelah di *forwarding* ke jenis layanan 2, maka nomor antrian 1 mendapatkan prioritas lebih tinggi dibandingkan nomor antrian yang lebih besar, sehingga nomor antrian 1 dilayani lebih dahulu dari pada nomor 9 dan 11.



Gambar 3.1 *Multiple Channel Queue* Dengan *Forwarding*

3.3. Analisis Metode Pembuatan Aplikasi

Pada bagian ini, penulis menjelaskan mulai dari blok diagram, *flowchart* dan metode pembuatan aplikasi.



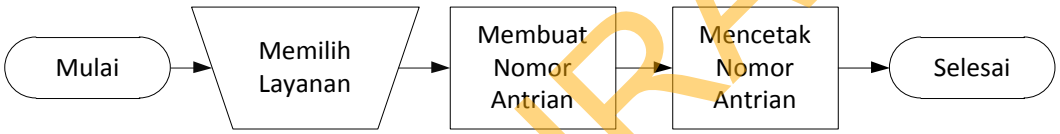
Gambar 3.2 Blok Diagram Cara Kerja aplikasi

Penjelasan blok diagram cara kerja aplikasi :

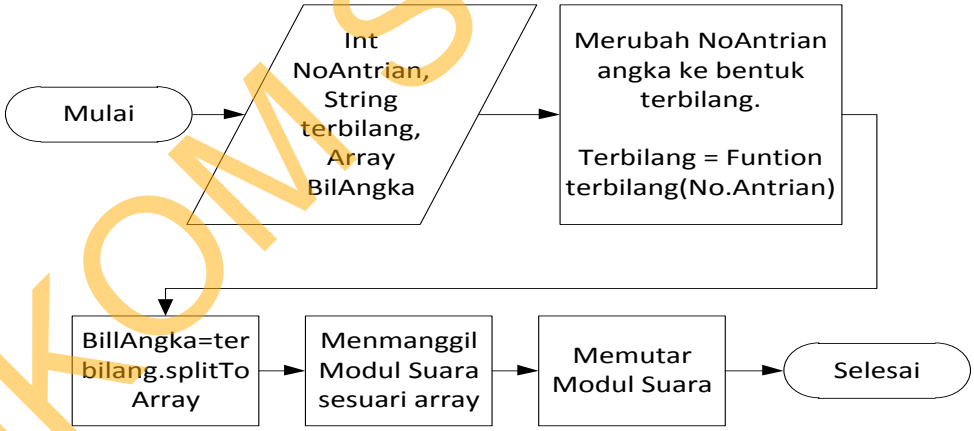
1. Komputer ambil nomor antrian dan mempu membuat nomor antrian yang kemudian dicetak oleh *printer* yang terhubung dengan komputer ambil nomor.
2. Nomor antrian akan tersimpan di komputer *server* untuk menunggu panggilan. Komputer *server* menampilkan nomor urut yang mendapat panggilan saat itu (nomor antrian terakhir) dan di *Customer Service* mana akan dilayani.

3. *Customer Service* akan memberi perintah pada aplikasi untuk memanggil nomor antrian berikutnya.
4. Komputer *server* akan merespon dan melakukan panggilan suara. Kemudian akan menampilkanya di monitor *display* nomor antrian.

Dari blok diagram pada gambar 3.2, aplikasi berjalan sebagai berikut dijelaskan dalam *flowchart*.



Gambar 3.3 *Flowchart* mengambil Nomor Antrian



Gambar 3.4 *Flowchart* memanggil Nomor Antrian

Dalam pengerjaan aplikasi, penulis melakukan langkah-langkah sebagai berikut:

1. Merekam *file* suara.

   File suara merupakan *file* yang akan diputar ketika *customer service* memanggil nomor antrian. *File-file* suara ini berupa angka-angka, bilangan, dan besaran bilangan sesuai dengan pengucapan bilangan dalam bahasa Indonesia. Bilangan tersebut antara lain:

   Tabel 3.1 Perencanaan *file* suara yang akan digunakan

Kata Angka/Bilangan	Kata Besaran Bilangan
Satu, Dua, Tiga, Empat, Lima, Enam, Tujuh, Delapan, Sembilan, Sepuluh, Sebelas, Seratus, Seribu	Belas, Puluh, Ratus, Ribu

   Dengan asumsi jumlah antrian per harinya hanya mencapai ratusan dan tidak mencapai ribuan. Namun masih disiapkan untuk kata "Ribu".

2. Membangun *database*.

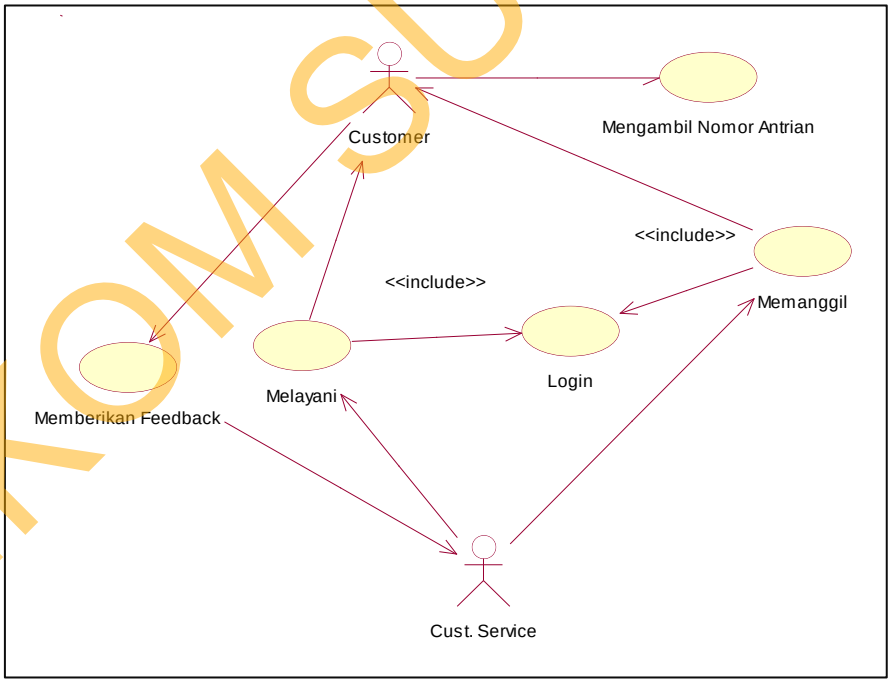
   Database yang dibangun akan digunakan untuk keperluan aplikasi dalam mencatat semua transaksi yang terjadi. *Database* akan dibuat berdasarkan desain ERD yang telah dijelaskan sebelumnya.

3. Membangun aplikasi.

   Selanjutnya adalah membangun aplikasi dengan menggunakan *file-file* suara dan *database* yang telah dibangun.

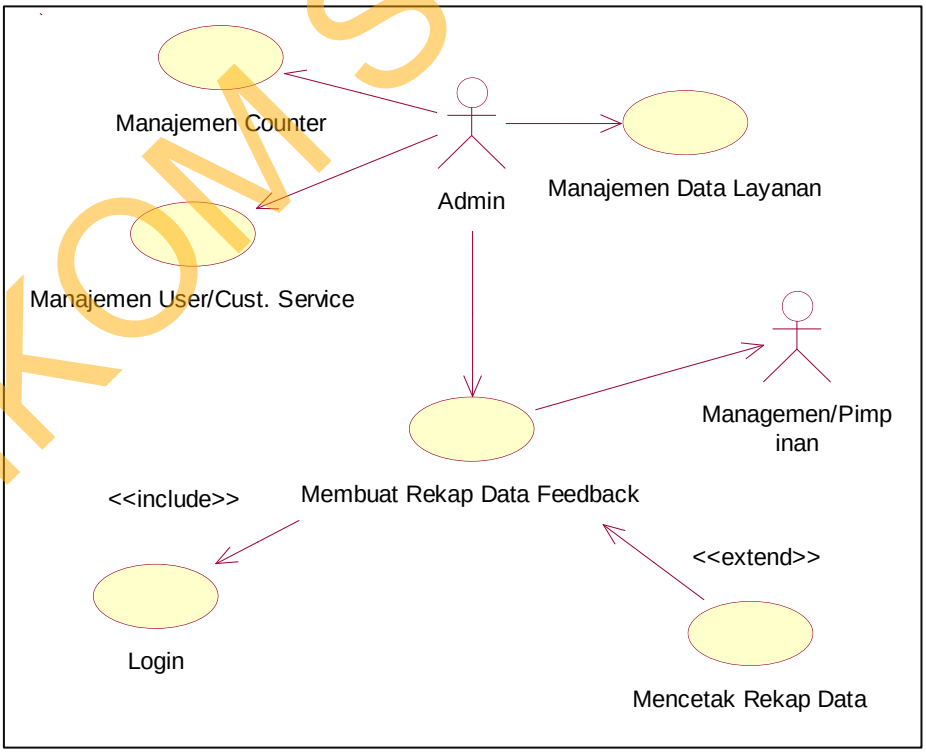
3.4. Use Case Diagram

Use case diagram digunakan untuk menggambarkan kemampuan atau kegunaan yang dimiliki aplikasi. *Use Case* diagram terdiri dari beberapa aktor dan *use case* yang saling berhubungan, yang menggambarkan kegunaan aplikasi. Berikut ini adalah *use case* diagram melayani antrian :



Gambar 3.5 *Use Case* Melayani Antrian

Pada gambar 3.5 *use case diagram* melibatkan 2 aktor, yaitu *Customer Service* dan *Customer*. *Customer* mengambil nomor antrian untuk bisa masuk ke *waiting line* sesuai dengan layanan yang diinginkan. *Customer Service* yang sebelumnya sudah melakukan *login* kemudian memanggil nomor antrian sesuai urutan antrian dan jenis layanan yang dilayani pada *counter* tersebut. Setelah dipanggil, *customer* yang bersangkutan akan datang ke *counter* untuk mendapatkan pelayanan. Setelah selesai, *Customer Service* akan mendapatkan *feedback* dari *customer* atas pelayanan yang diberikan dan disimpan. Selanjutnya adalah use case *administrator* :

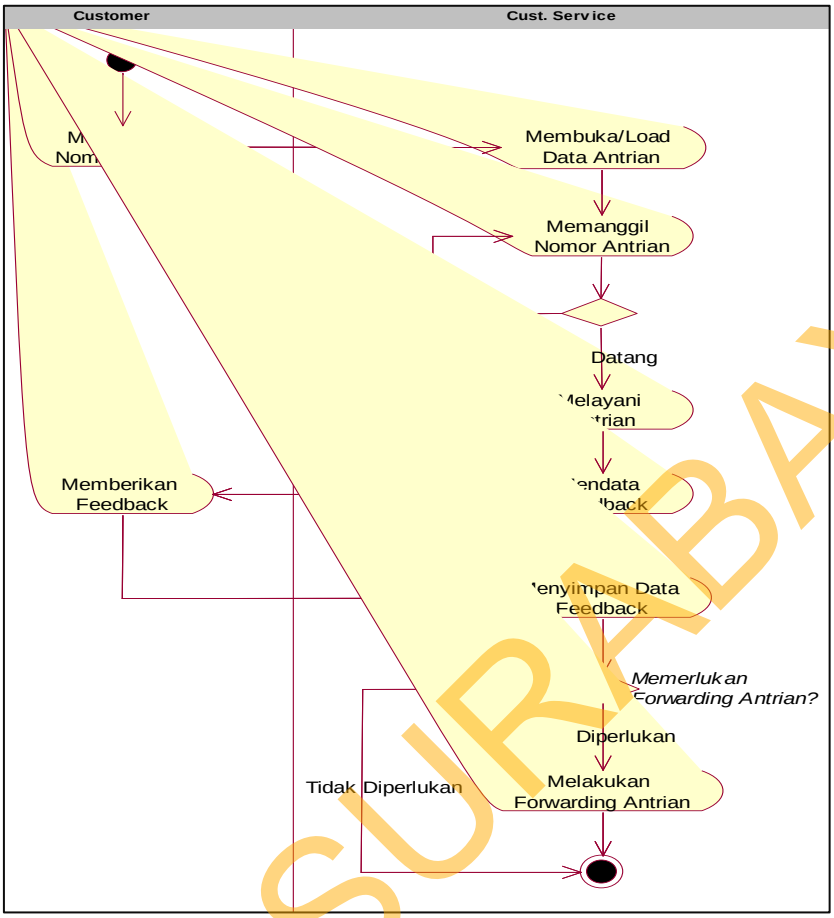


Gambar 3.6 *Use Case Administrator*

Pada gambar 3.6 *use case Administrator* melibatkan 2 aktor yaitu *Admin* dan Manajer/Pimpinan. Dimana *admin* dapat melakukan manajemen *counter*, yaitu menambah, mengurangi dan mengatur ulang *counter* yang sudah ada sesuai dengan keperluan perusahaan. Manajemen data layanan, yaitu menambah, mengurangi dan mengatur jenis dan jumlah layanan yang ada yang digunakan dalam sistem antrian. Manajemen *User*/*Customer Service*, yaitu mendaftarkan user baru dan mengedit data *user*/*Customer Service* yang nantinya akan bertugas. Membuat dan mencetak rekap data pelayanan, yaitu membuat rekap data tentang kepuasan *Customer* terhadap layanan dari *Customer Service* yang kemudian diserahkan kepada Manajer/Pimpinan.

3.5. Activity Diagram

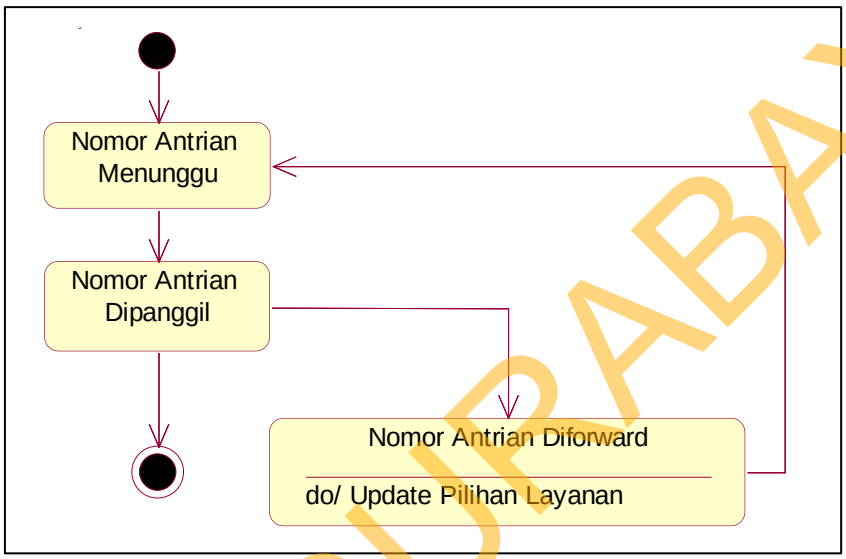
Activity diagram menggambarkan alur jalanya aplikasi berupa aktifitas-aktifitas yang dilakukan oleh aktor yang telah digambar pada *use care* diagram pada sub bab sebelumnya. Berikut ini adalah *activity diagram* melayani antrian :



Gambar 3.7 *Activity Diagram* Melayani Antrian

Aktivitas pada gambar 3.7 dimulai ketika *customer* mengambil nomor antrian, setelah data di *load* kemudian *Customer Service* memanggil nomor antrian. Jika nomor yang dipanggil tidak datang makan *Customer Service* akan memanggil nomor selanjutnya. Jika *customer* yang bersangkutan datang, maka dilanjutkan melayani *customer*. Setelah selesai melayani, *customer* akan memeberikan *feedback* kepada *Customer Service*. Selanjutnya, jika perlu melakukan *forwarding*, maka nomor antrian tersebut akan diforward ke layanan

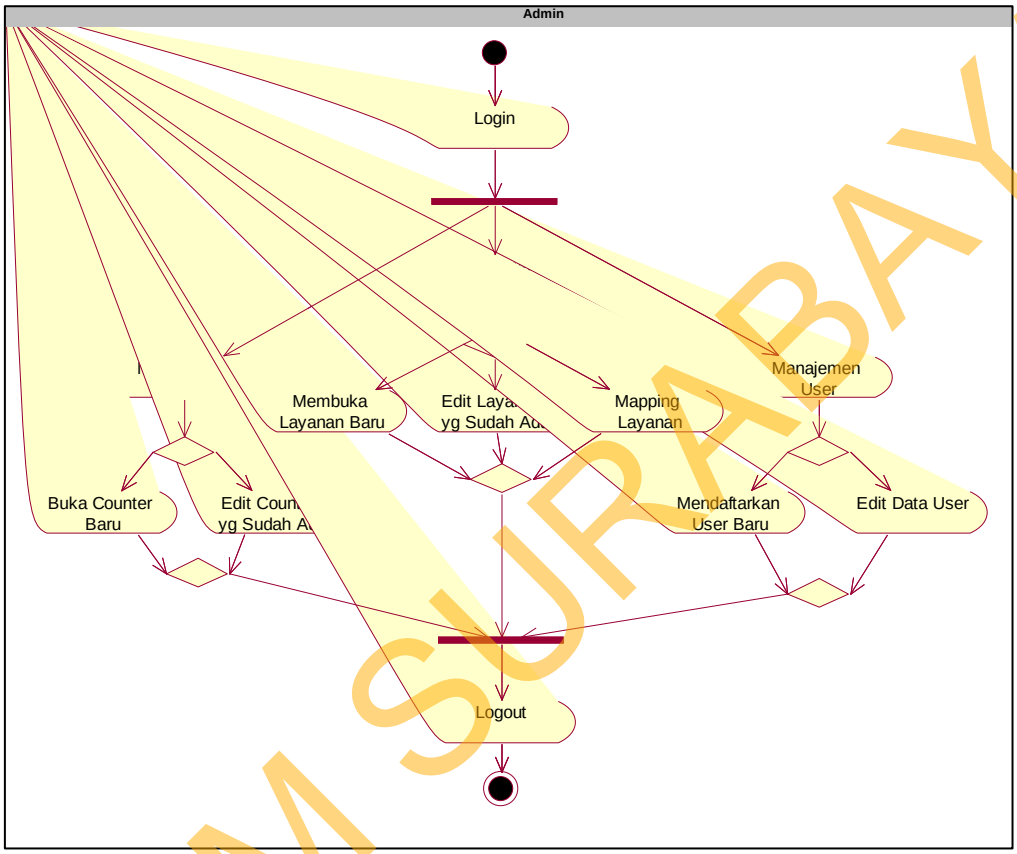
lainya. Ada beberapa kondisi yang dialami oleh nomor antrian yang digambarkan dalam *statechart* diagram nomor antrian pada gambar 3.8 berikut :



Gambar 3.8 *Statechart Diagram* Nomor Antrian

Nomor menunggu adalah keadaan dimana nomor antrian yang sudah diambil namun belum dipanggil oleh *operator/Customer Service* untuk mendapatkan pelayanan. Nomor antrian dipanggil adalah nomor antrian yang sebelumnya menunggu dan telah dipanggil oleh *operator*/*Customer Service* untuk dilayani. Kondisi ini termasuk diantaranya nomor yang dipanggil tapi tidak datang. Setelah dipanggil, nomor antrian bisa diforward atau tidak. Jika diforward, maka nomor antrian akan memiliki status diforward yang kemudian statusnya sama dengan nomor antrian menuggu yang menunggu untuk

dipanggil oleh operator. Selanjutnya adalah *activity diagram* administrator :



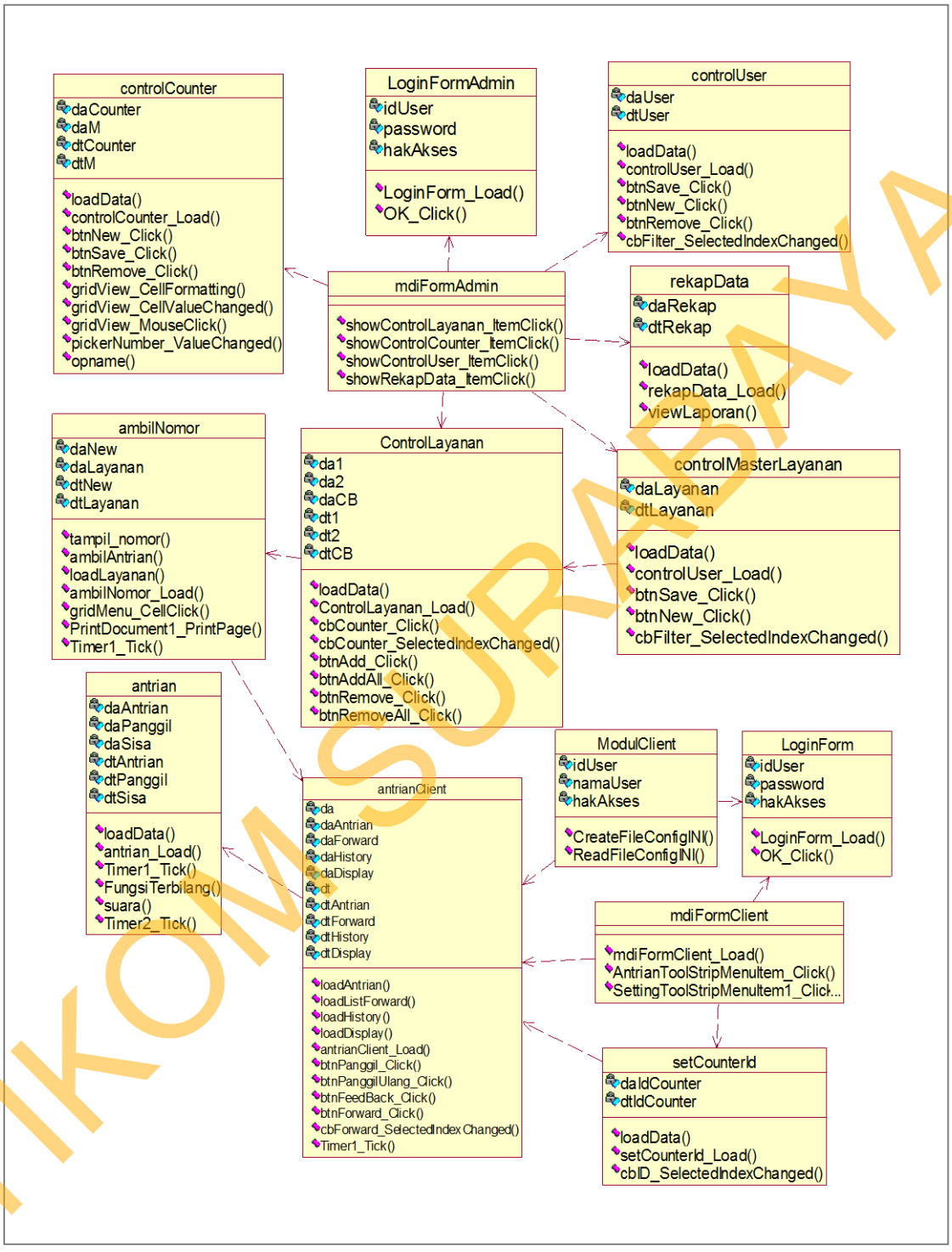
Gambar 3.9 *Activity Diagram Administrator*

Pada gambar 3.9, admin melakukan administrasi terhadap data dengan *login* terlebih dahulu. Administrasi data yang dapat dilakukan oleh *admin* antara lain, Manajemen *counter*, jumlah dan jenis *counter* dapat disesuaikan dengan kebutuhan. Setiap *counter* yang telah dibuat dapat disesuaikan lagi sesuai dengan keinginan. Manajemen layanan, jumlah dan jenis layanan dapat disesuaikan dengan menambah

layanan baru, mengurangi layanan dan mengatur ulang layanan yang sudah ada. Manajemen *user*, *admin* dapat menambah/mendaftarkan *user* baru jika terjadi penambahan karyawan dan perubahan data karyawan.

3.6. Class Diagram

Class diagram menggambarkan hubungan antar *class-class* yang ada secara statik. Dalam *class diagram* ini dibagi menjadi 2 jenis kegunaan utama yaitu untuk adminirtrasi data dan untuk melayani antrian. Berikut ini adalah *class* diagram aplikasi :



Gambar 3.10 *Class Diagram* Aplikasi Sistem Antrian

3.6.1. Administrasi Data

Merupakan proses pengolahan data, antara lain : data user, *counter* dan layanan. Modul ini menyediakan *method-method* yang berhubungan dengan pengolahan data *user*, *counter* dan layanan.

Tabel 3.2 *Class* mdiFormAdmin

mdiFormAdmin	
Kegunaan	Sebagai *form* mdi pada proses administrasi data
Dependency	-
Class Induk	-
Atribut	**Kegunaan**
-	-
Method	**Kegunaan**
showControlLayanan_ItemClick	Memberikan *event* kepada *button* showControlLayanan untuk memanggil *form* ControlLayanan
showControlCounter_ItemClick()	Memberikan *event* kepada *button* showControlCounter untuk memanggil *form* ControlCounter
showControlUser_ItemClick	Memberikan *event* kepada *button* showControlUser untuk memanggil *form* ControlUser
showRekapData_ItemClick	Memberikan *event* kepada *button* showRekapData untuk memanggil *form* RekapData

Tabel 3.3 *Class* LoginFormAdmin

LoginFormAdmin	
Kegunaan	Sebagai *form login* untuk hak akses aplikasi
Dependency	mdiFormAdmin, ModulAdmin
Class Induk	-
Atribut	**Kegunaan**
idUser	Sebagai id untuk mendapat akses aplikasi
Password	*Password*/kata sandi
hakAkses	Hak akses yang dimiliki. Dalam hal ini hanya hak akses *admin* yang diberikan akses.
Method	**Kegunaan**
LoginForm_Load	Konfigurasi awal saat *form* di load
OK_Click	Menjalankan *query login* dan mencocokan *username* dan *password* yang dimasukan.

Tabel 3.4 *Class* controlMasterLayanan

controlMasterLayanan	
Kegunaan	Sebagai *form* untuk proses administrasi data layanan
Dependency	mdiFormAdmin
Class Induk	-
Atribut	**Kegunaan**
daLayanan	Data *adapter* untuk semua data layanan dari tabel layanan
dtLayanan	*Datatable* untuk menampung data dari daLayanan
Method	**Kegunaan**
loadData	Mengambil semua data yang diperlukan dari *database*
controlMasterLayanan _Load	Memanggil *method* loadData()pada saat *form Load*.
btnNew_Click	Menambahkan satu *record* baru untuk data layanan.
btnSave_Click	Menyimpan perubahan yang dilakukan oleh *admin* terhadap data layanan.
btnRemove_Click	Menghapus satu *record* data layanan
cbFilter_SelectedIndexChanged	Memfilter *record* yang sudah tersimpan.

Tabel 3.5 *Class* controlCounter

controlCounter	
Kegunaan	Sebagai *form* untuk proses administrasi data *counter*
Dependency	mdiFormAdmin
Class Induk	-
Atribut	**Kegunaan**
daCounter	*Dataadapter* untuk semua data *counter* dari tabel *counter*
daM	*Dataadapter* untuk semua *data* counter dari tabel *counter monitoring*
dtCounter	*Datatable* untuk semua data *counter* dari tabel *counter*
dtM	*Datatable* untuk semua data *counter* dari tabel *counter monitoring*
Method	**Kegunaan**
loadData	Mengambil semua data yang diperlukan dari *database*
controlCounter_Load	Memanggil *method* loadData()pada saat *form* Load.
btnNew_Click	Menambahkan satu *record* baru untuk data *counter*.
btnSave_Click	Menyimpan perubahan yang dilakukan oleh *admin* terhadap data *counter*.
btnRemove_Click	Menghapus satu *record* data *counter*
gridView_CellFormatting	Mengatur *layout datagridview* yang digunakan untuk menampilkan data *counter*.
gridView_CellValueChanged	Menjalankan *event* ketika terjadi perubahan terhadap isi *cell* dari *datagridview* untuk mencegah *entry* data yang tidak valid.
gridView_MouseClick	Menjalankan *event* ketika terjadi perubahan terhadap isi *cell* dari *datagridview* untuk mencegah *entry* data yang tidak valid.
pickerNumber_ValueChanged	Sebagai validasi *input* data angka ke dalam sebuah *cell*.

Tabel 3.6 *Class* ControlLayanan

ControlLayanan	
Kegunaan	Sebagai *form* untuk proses *mapping* layanan
Dependency	mdiFormAdmin, controlMasterLayanan
Class Induk	-
Atribut	**Kegunaan**
da1	Sebagai data *adapter* untuk data layanan yang belum di-*mapping*
da2	Sebagai *datatable* untuk data layanan yang belum di-*mapping*
daCb	Sebagai *dataadapter* untuk data *counter* yang akan ditampilkan di *combobox*
dt1	Sebagai *dataadapter* untuk data layanan yang sudah di-*mapping*
dt1	Sebagai *datatable* untuk data layanan yang sudah di-*mapping*
dtCb	Sebagai *datatable* untuk data *counter* yang akan ditampilkan di *combobox*
Method	**Kegunaan**
loadData	Mengambil semua data yang diperlukan dari *database*
ControlLayanan_Load	Memanggil *method* loadData()pada saat *form* Load danmelakukan *fill* data ke kontrol-kontrol pada *form*.
cbCounter_Click	Mengambil data dari *database* untuk ditampilkan di cbCounter.
cbCounter_SelectedIndex Changed	Menjalankan *event* ketika terjadi perubahan pada *combobox* cbCounter.
btnAdd_Click	Menambahkan/*mapping* 1 layanan ke *counter* tertentu
btnAddAll_Click	Menambahkan/*mapping* semua layanan ke *counter* tertentu
btnRemove_Click	Menghilangkan 1 layanan dari *counter* tertentu
btnRemoveAll_Click	Menghilangkan semua layanan ke *counter* tertentu

Tabel 3.7 *Class* controlUser

controlUser	
Kegunaan	Sebagai *form* untuk proses administrasi *user*
Dependency	mdiFormAdmin
Class Induk	-
Atribut	**Kegunaan**
daUser	*Dataadapter* untuk semua data *user*
dtUser	*Datatable* untuk menampung hasil dari daUser
Method	**Kegunaan**
loadData	Mengambil semua data yang diperlukan dari *database*
controlUser_Load	Memanggil *method* loadData()pada saat form Load.
btnNew_Click	Menambahkan satu *record* baru untuk data *counter*.
btnSave_Click	Menyimpan perubahan yang dilakukan oleh *admin* terhadap data *counter*.
btnRemove_Click	Menghapus satu *record* data *counter*
cbFilter_SelectedIndexChanged	Memfilter *record* yang sudah tersimpan.

Tabel 3.8 *Class* rekapData

rekapData	
Kegunaan	Sebagai *form* untuk proses administrasi rekap data
Dependency	mdiFormAdmin
Class Induk	-
Atribut	**Kegunaan**
daRekap	*Dataadapter* untuk semua data rekap hasil melayani *customer*
dtRekap	*Datatable* untuk menampung hasil dari daRekap
Method	**Kegunaan**
loadData	Mengambil semua data yang diperlukan dari *database*
rekapData_Load	Memanggil *method* loadData()pada saat *form* Load.
viewLaporan	Melihat laporan rekap data

Setiap *class* yang disebutkan diatas memudahkan *admin* untuk melakukan pengaturan *counter* dan konfigurasi layanan serta membuat laporan jika diperlukan oleh manajer/pimpinan. *Class* mdiFormAdmin pada tabel 3.2 merupakan *form* induk yang digunakan

untuk memanggil *form-form children* di bawahnya. Sehingga memudahkan bernavigasi dari satu *form* ke form lainya. Class LoginFormAdmin pada tabel 3.3 berguna sebagai hak akses ke aplikasi.

Class ControlMasterLayanan pada tabel 3.4 berguna untuk menambah, meghapus dan *mapping* layanan ke *counter*, sehingga dapat diatur sebuah *counter* dapat melayani hanya satu jenis layanan atau berbagai layanan. *Class* controlCounter pada tabel 3.5 merupakan *form* untuk mengontrol *counter* yang telah dibuat dan tersimpan ke tabel master counter. *Admin* dapat mengubah jumlah dan nama *counter* sesuai dengan kebutuhan memalui *class* ini. *Class* ControlLayanan pada tabel 3.6 mengatur jenis dan jumlah layanan yang diperlukan dan melanjutkan kerja dari *class* controlCounter pada tabel 3.5 untuk melakukan *mapping* layanan yang telah dibuat ke *counter* tertentu dimana *admin* dapat mengatur layanan apa saja yang dilayani di sebuah *counter*. *Class* controlUser pada tabel 3.7 digunakan untuk mengatur dan menambah *user* yang boleh mengakses aplikasi, ini diperlukan jika terjadi penambahan atau pergantian pegawai.

3.6.2. Melayani Antrian

Merupakan proses mulai dari mengambil nomor antrian, memanggil, menerima *feedback*, hingga melakukan *forwarding*.

Tabel 3.9 *Class* ambilNomor

ambilNomor	
Kegunaan	Sebagai *form* untuk mengambil nomor antrian yang dilakukan oleh *customer*
Dependency	ControlLayanan
Class Induk	-
Atribut	**Kegunaan**
daNew	*Dataadapter* untuk nomor antrian yang baru
daLayanan	*Dataadapter* untuk semua jenis layanan yang tersedia
dtNew	*Datatable* untuk menampung hasil dari daNew
dtLayanan	*Datatable* untuk menampung hasil dari daLayanan
Method	**Kegunaan**
tampil_nomor	Menampilkan nomor antrian yang diambil oleh *customer* pada *form* ambilNomor
ambilAntrian	Mengambil *list* nomor antrian yang sudah diambil untuk menghitung nomor antrian yang akan diambil oleh *customer* selanjutnya
loadLayanan	Mengambil semua data yang diperlukan dari *database*
ambilNomor_Load	Menjalankan semua *method* yang dibutuhkan saat program *startup*
gridMenu_CellClick	Menghasilkan nomor antrian berdasarkan jenis layanan yang dipilih
PrintDocument1_PrintPage	Mencetak nomor antrian
Timer1_Tick	Merefresh data dari *database*

Tabel 3.10 *Class* mdiFormClient

mdiFormClient	
Kegunaan	Sebagai *form* mdi untuk *counter*/Customer Service
Dependency	-
Class Induk	-
Atribut	**Kegunaan**
-	-
Method	**Kegunaan**
mdiFormClient_Load	Konfigurasi awal *form*
AntrianToolStripMenuItem_Click	Memanggil *form* untuk melayani antrian
SettingToolStripMenuItem1_Click	Memanggil *form setting*

Tabel 3.11 *Class* ModulClient

ModulClient	
Kegunaan	Sebagai modul untuk menampung variabel *login* pada *form login* dan *client*
Dependency	-
Class Induk	-
Atribut	**Kegunaan**
idUser	Sebagai id untuk mendapat akses aplikasi
Password	*Password*/kata sandi
hakAkses	Hak akses yang dimiliki.
Method	**Kegunaan**
createFileConfigINI	Membuat *file* 'konfigurasi.ini' yang berguna untuk menyimpan konfigurasi/*setting* program
readFileConfigINI	Membaca *file* 'konfigurasi.ini' yang berguna untuk menerapkan konfigurasi/*setting* yang telah dibuat

Tabel 3.12 *Class* antrianClient

antrianClient	
Kegunaan	Sebagai *form* untuk memanggil nomor antrian
Dependency	mdiFormClient, ModulClient
Class Induk	-
Atribut	**Kegunaan**
da	*Dataadapter* yang menjadi *adapter* untuk memanipulasi data nomor antrian
daAntrian	*Dataadapter* yang berfungsi sebagai *queue* dari nomor antrian yang menunggu untuk dipanggil
daForward	*Dataadapter* untuk mengakses data layanan yang ada untuk keperluan *forward* nomor antrian
daHistory	*Dataadapter* yang menjadi *adapter* untuk memanipulasi data di tabel historyMelayani
daDisplay	*Dataadapter* yang menjadi *adapter* untuk memanipulasi data di tabel counterMonitoring
dt	*Datatable* untuk menampung data dari da
dtAntrian	*Datatable* untuk menampung data dari daAntrian
dtForward	*Datatable* untuk menampung data dari daForward
dtHistory	*Datatable* untuk menampung data dari daHistory
dtDisplay	*Datatable* untuk menampung data dari daDisplay
Method	**Kegunaan**
loadAntrian	Mengambil data nomor antrian dari tabel antrian
loadListForward	Mengambil data jenis layanan untuk keperluan *forwarding* antrian ke jenis layanan tertentu
loadHistory	Mengambil data nomor antrian dari tabel historyMelayani
loadDisplay	Mengambil data nomor antrian dari tabel counterMonitoring untuk *record form* yang bersangkutan
Method	**Kegunaan**
antrianClient_Load	Menjalankan *method* loadAntrian, loadListForward, loadHistory dan loadDisplay saat *form startup*
btnPanggil_Click	Memanggil nomor antrian dan memerintahkan *form* antrian untuk melakukan panggilan suara
btnPanggilUlang_Click	Mengulangi panggilan pada saat panggilan terhadap nomor tertentu, dimana nomor yang dipanggil masih sama dengan nomor yang dipanggil oleh *method* btnPanggil_Click
btnFeedBack_Click	Menerima *input feedback* dari *customer*
btnForward_Click	Melakukan *forwarding* ke nomor antrian tertentu sesuai dengan *combobox forwarding*
cbForward_SelectedIndex Changed	Memilih pilihan tujuan *forwarding* menuju ke jenis antrian tertentu
Timer1_Tick	Merefresh data

Tabel 3.13 *Class* LoginForm

LoginForm	
Kegunaan	Sebagai *form login* untuk hak akses aplikasi
Dependency	mdiFormClient, ModulClient
Class Induk	-
Atribut	**Kegunaan**
idUser	Sebagai id untuk mendapat akses aplikasi
Password	*Password*/kata sandi
hakAkses	Hak akses yang dimiliki.
Method	**Kegunaan**
LoginForm_Load	Konfigurasi awal saat *form* di load
OK_Click	Menjalankan *query login* dan mencocokan *username* dan *password* yang dimasukan.

Tabel 3.14 *Class* setCounterId

setCounterId	
Kegunaan	Sebagai form untuk mengkonfigurasi form antrianClient
Dependency	-
Class Induk	-
Atribut	**Kegunaan**
daIdUser	*Dataadapter* untuk mengakses data ID *user*
dtIdUser	*Datatable* untuk menampung data yang dihasilkan oleh daIdUser
Method	**Kegunaan**
loadData	Meload semua data yang dibituhkan dari *database*
setCounterId_Load	Menjalankan *method* loadData saat *form startup*
cbID_SelectedIndexChanged	*Mapping* form/memberi ID pada *form* yang bersangkutan sesuai dengan yang terdaftar di *database*

Tabel 3.15 *Class* antrian

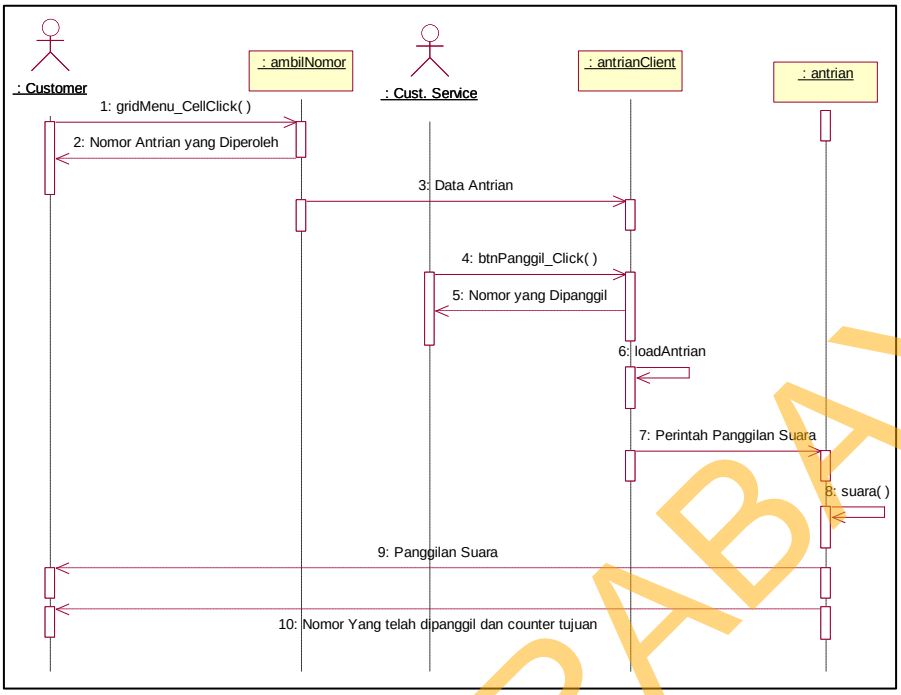
antrian	
Kegunaan	Sebagai *form* untuk melakukan panggilan suara dan menampilkan nomor yang dipanggil oleh form antrianClient
Dependency	antrianClient
Class Induk	-
Atribut	**Kegunaan**
daAntrian	*Dataadapter* untuk mengambil nomor antrian dari tabel antrian.
daPanggil	*Dataadapter* untuk mengambil nomor antrian yang sudah dipanggil dari tabel antrian.
daSisa	*Dataadapter* untuk mengambil sisa nomor antrian yang menunggu.
dtAntrian	*Datatable* untuk menampung data dari daAntrian
dtPanggil	*Datatable* untuk menampung data dari daPanggil
dtSisa	*Datatable* untuk menampung data dari daSisa
Method	**Kegunaan**
loadData	Meload semua data yang dibituhkan dari *database*
antrian_Load	Menjalankan *method* loadData saat *form startup*
Timer1_Tick	Merefresh data dan data yang ditampilkan
FungsiTerbilang	Menerjemahkan nomor antrian menjadi teks berupa kalimat terbilang dari nomor antrian tersebut
suara	Merubah teks terbilang dari sebuah nomor antrian dan nomor *counter* menjadi *array* untuk melakukan panggilan suara dengan memainkan file suara sesuai *array* yang merujuk ke nama *file* dengan format "xxxx.wav"
Timer2_Tick	Mengatur pemutaran *file* suara agar tidak terjadi tabrakan/menindih dengan memastikan bahwa panggilan sebelumnya telah diselesaikan sebelum melakukan panggilan selanjutnya

Customer mengambil nomor di form ambilNomor yang ditampilkan pada tabel 3.9 yang kemudian disimpan ke dalam *database* sesuai dengan jenis layanan yang diambilnya. *Class* mdiFormClient pada tabel 3.10 membantu navigasi *Customer Service* dari satu *form* ke *form* lainya. Class ModulClient pada tabel 3.11 menyimpan informasi *login* selama aplikasi berjalan. *Class*

antrianClient tabel 3.12 digunakan untuk memanggil nomor antrian yang dilakukan oleh *Customer Service*. Class LoginForm pada tabel 3.13 menjadi akses masuk ke aplikasi. Class setCounterId tabel 3.14 untuk mengatur/*mapping counter* ke *id counter* tertentu sebagai identitas *counter* tersebut, class ini hanya bisa diakses oleh *admin*. Class antrian pada tabel 3.15 melakukan panggilan suara dan menampilkan nomor antrian yang dipanggil serta *counter* yang dituju.

3.7. Sequence Diagram

Sequence diagram menggambarkan interaksi antar *class* dan *object* dengan saling mengirik *message*/*method* antara satu dengan lainya. Alur *sequence diagram* dibaca mulai dari atas kebawah. Berikut ini adala *sequence* diagram aplikasi selama melayani antrian.



Gambar 3.11 *Sequence Diagram*

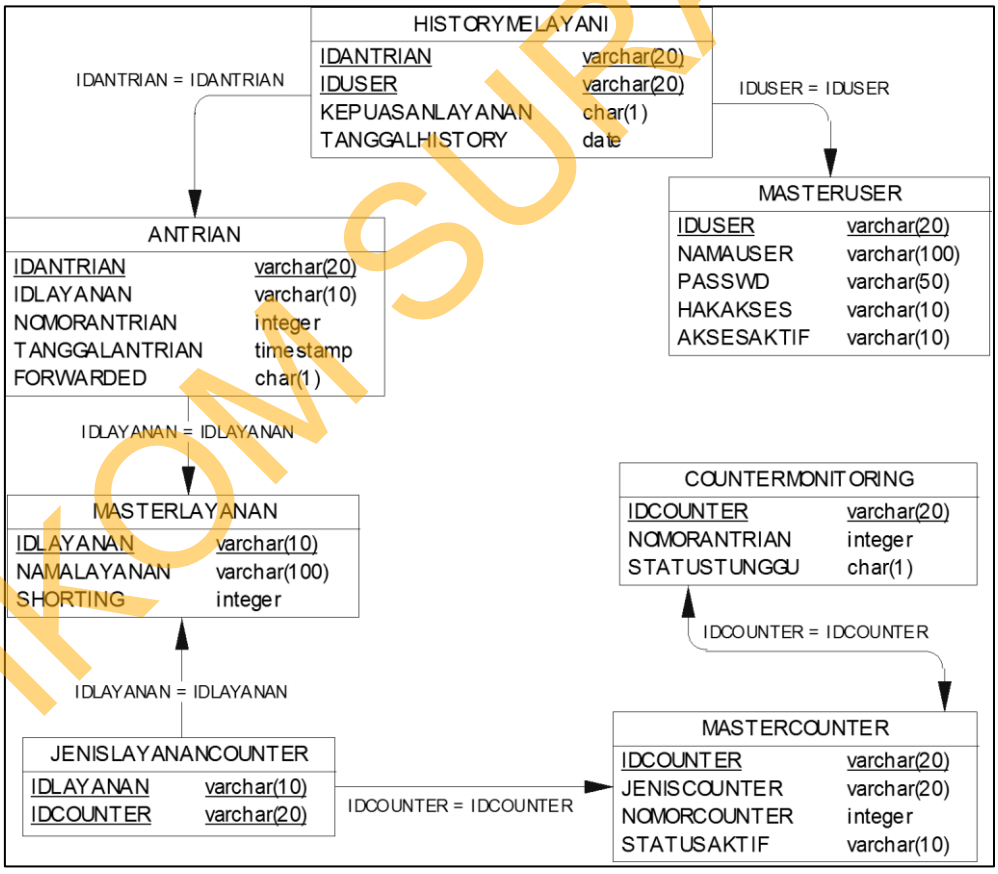
Alur mulai dari *customer* yang datang dan mengambil nomor antrian. Data antrian kemudian di ambil dari *database* oleh *class* antrianClient. *Customer Service* kemudian memanggil nomor dengan *method* btnPanggil_click() pada *class* antrianClient dan mendapatkan respon berupa nomor antrian yang dipanggil. *Class* antrianClient kemudian meneruskan perintah berupa perintah panggilan suara lewat *database* kepada *calss* antrian, kemudian *class* antrian menjalankan method suara untuk melakukan panggilan suara dan mengirimkan respon berupa panggilan suara dan tampilan nomor antrian kepada *customer*.

3.8. Perancangan Database

Pada bagian rancangan *database* ini akan dijelaskan rancangan struktur database mulai dari ERD (*Entity Relationship Diagram*) hingga ke struktur tabel.

3.8.1. ERD (Entity Relationship Diagram)

Berikut ini adalah ERD dalam bentuk PDM (*Physical Data Model*).



Gambar 3.12 *Physical Data Model*

3.8.2. Struktur Database

Bagian ini akan menjelaskan struktur database yang akan dibuat. Database yang akan dibuat memiliki 3 tabel master yaitu masterCounter, masterLayanan dan masterUser. Serta 4 tabel transaksi yaitu tabel jenisLayananCounter, antrian, historyMelayani dan counterMonitoring.

1. Nama Tabel : masterCounter

   Fungsi : menyimpan data *counter* yang aktif

   Primary Key : idCounter

Tabel 3.16 Struktur tabel masterCounter

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idCounter	varchar	20	Identitas *counter*
2	jenisCounter	varchar	20	Jenis *counter*
3	nomorCounter	integer		Nomor urut
4	statusAktif	varchar	10	Status keaktifan *counter*

2. Nama Tabel : masterLayanan

   Fungsi : menyimpan data layanan yang ada

   Primary Key : idLayanan

Tabel 3.17 Struktur tabel masterLayanan

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idLayanan	varchar	10	Id layanan
2	namaLayanan	varchar	100	Nama layanan
3	Shorting	integer		Urutan untuk mendisplay layanan

3. Nama Tabel : masterUser

   Fungsi : menyimpan data *user*

   Primary Key : idUser

Tabel 3.18 Struktur tabel masterUser

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idUser	varchar	20	Id Login *user*
2	namaUser	varchar	100	Nama Lengkap *User*
3	passwd	varchar	50	*Password* untuk *login*
4	hakAkses	varchar	10	Hak akses yang dimiliki *user*
5	aksesAktif	varchar	10	Menunjukan apakah akun *user* aktif atau tidak

4. Nama Tabel : jenisLayananCounter

   Fungsi : memetakan layanan ke *counter*.

   Primary Key : -

   Foreign Key : idCounter, idLayanan

Tabel 3.19 Struktur tabel jenisLayananCounter

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idCounter	varchar	20	FK dari tabel masterCounter
2	idLayanan	varchar	10	FK dari tabel masterLayanan

5. Nama Tabel : antrian

   Fungsi : menyimpan nomor antrian yang dibuat

   Primary Key : idAntrian

   Foreign Key : idLayanan

Tabel 3.20 Struktur tabel antrian

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idAntrian	varchar	20	Id Nomor Antrian
2	idLayanan	varchar	10	Id dari jenis layanan, FK dari tabel masterLayanan
3	nomorAntrian	integer		Nomor antrian
4	tanggalAntrian	datetime		Tanggal diambil dan berlakunya nomor antrian
5	forwarded	char	1	Menunjukan apakah nomor antrian diforward atau tidak

6. Nama Tabel : historyMelayani

   Fungsi : menyimpan data *user*

   Primary Key : -

   Foreign Key : idUser, idAntrian

Tabel 3.21 Struktur tabel historyMelayani

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idUser	varchar	20	Id user FK dari tabel masterUser
2	idAntrian	varchar	20	idAntrian FK dari tabel antrian
3	kepuasanLayanan	char	1	Kepuasan layanan
4	tanggalHistory	datetime		Tanggal antrian dilayani

7. Nama Tabel : counterMonitoring

   Fungsi : menyimpan data *user*

   Primary Key : -

   Foreign Key : idCounter

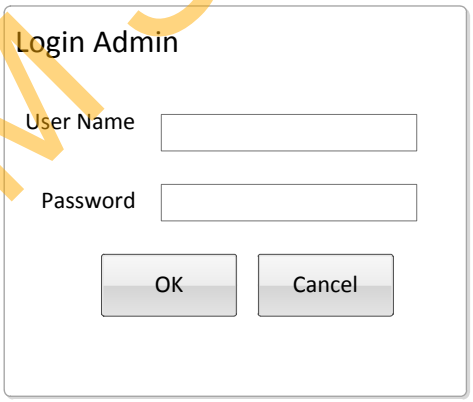
Tabel 3.22 Struktur tabel counterMonitoring

No	Nama Kolom	Tipe	Panjang	Keterangan
1	idCounter	varchar	20	Id counter FK dari tabel masterCounter
2	nomorAntrian	integer		Nomor antrian yang sedang dilayani pada counter tersebut
3	statusTunggu	varchar	50	Status tunggu untuk mengetahui apakah nomor antrian tersebut perlu dipanggil atau tidak

3.9. Desain Input/Output

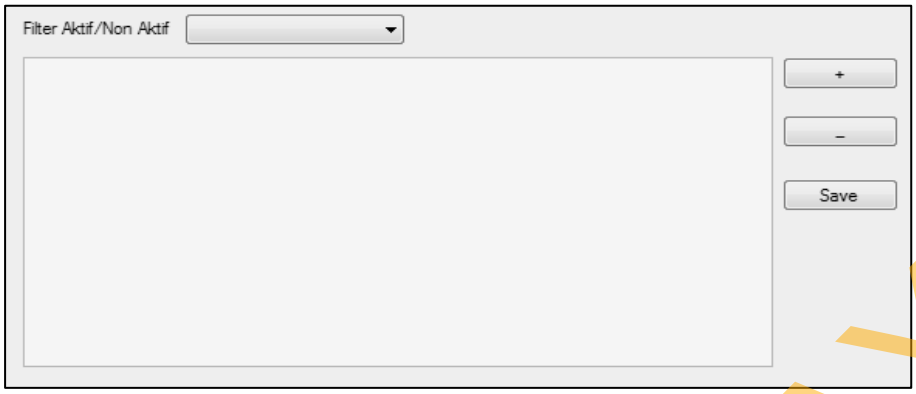
Selanjutnya akan dijelaskan desain *input* dan *output* dari aplikasi.

3.9.1. Administrasi data



Gambar 3.13 desain *form login admin*

Gambar 3.13 adalah desain *form login* untuk *admin*, sebagai hak akses masuk ke aplikasi.



Gambar 3.14 Desain *form* Master *User*

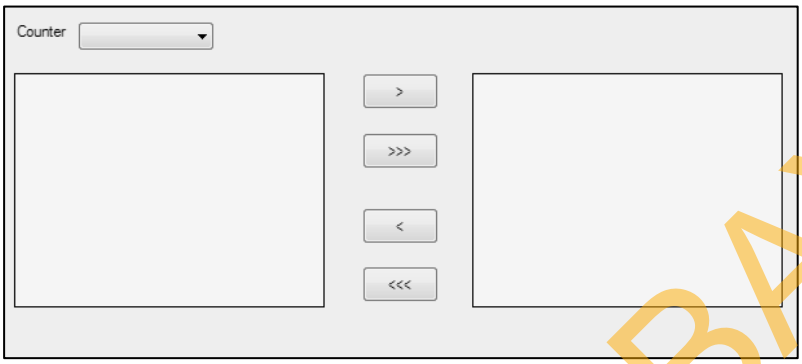
Gambar 3.14 adalah form controlUser untuk mendaftarkan *user* baru, mengedit atau menghapus *user*.

Gambar 3.15 desain form Master Layanan

Gambar 3.15 adalah *form* controlMasterLayanan untuk menambah, menghapus dan mengedit layanan.

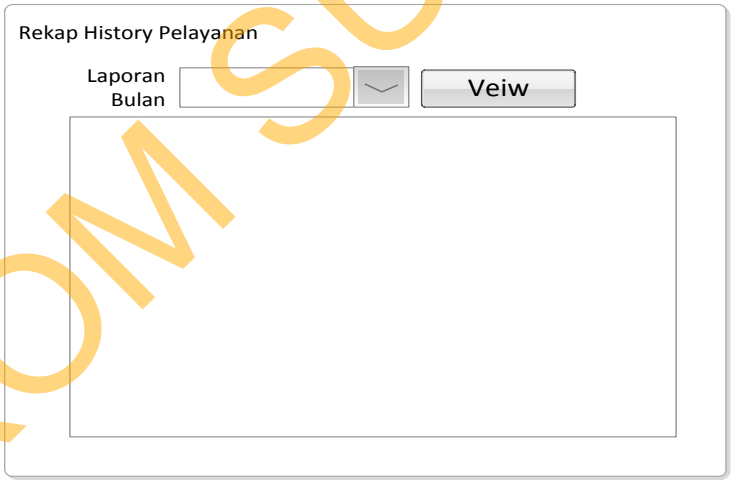
Gambar 3.16 Desain *form* controlCounter

Gambar 3.16 adalah desain *form* controlLayanan untuk menambah, menghapus dan mengedit *counter*.



Gambar 3.17 Desain *form* controlLayanan

Gambar 3.17 adalah desain *form* controlLayanan untuk mapping layanan ke *counter* yang diinginkan.



Gambar 3.18 Desain *form* rekapData

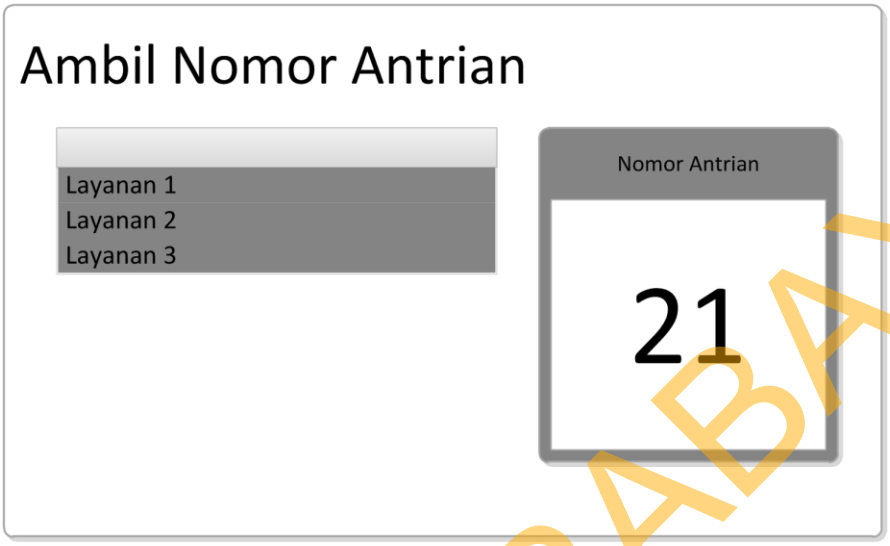
Gambar 3.18 adalah desain *form* rekapData yang berfungsi untuk melakukan rekap data pelayanan. *Combobox* Laporan Bulan berfungsi untuk memfilter data berdasarkan bulan tertentu. Terdapat sebuah

Datagridview berfungsi untuk menampilkan rekap data yang diinginkan. Tombol *view* berfungsi untuk melihat rekap dalam bentuk *report*.

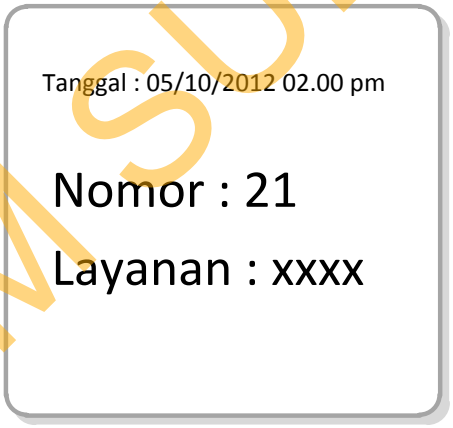
Gambar 3.19 Desain *Output Report* rekap data

Gambar 3.19 adalah desain *Output Report* rekap data yang dihasilkan saat menekan tombol *view* pada form rekapData pada gambar 3.20

3.9.2. Ambil Nomor Antrian



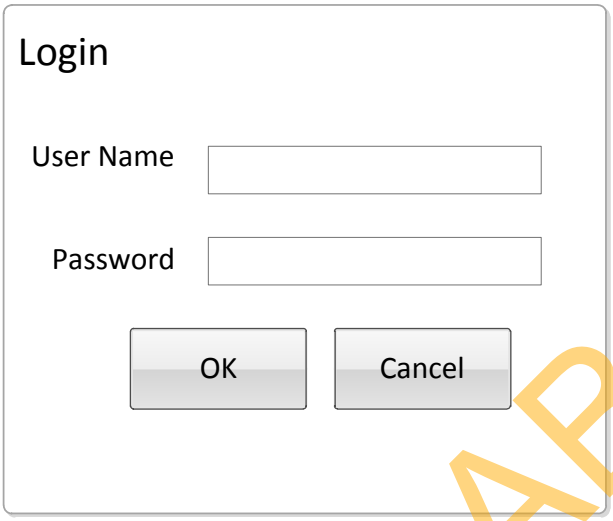
Gambar 3.20 Desain *form* ambilNomor



Gambar 3.21 Desain Nomor antrian yang dicetak

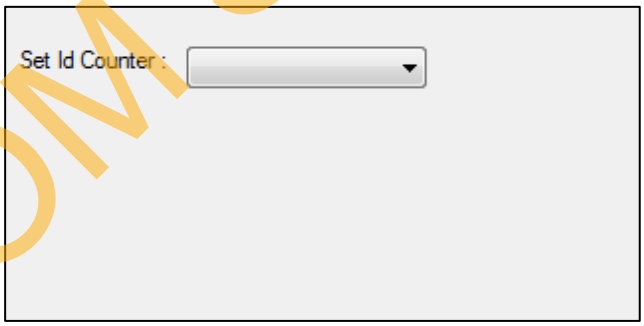
Gambar 3.20 adalah *form* ambilNomor berfungsi untuk mengambil nomor antrian. Kemudian nomor antrian akan dicetak oleh *printer* seperti gambar 3.21.

3.9.3. Memanggil Nomor Antrian



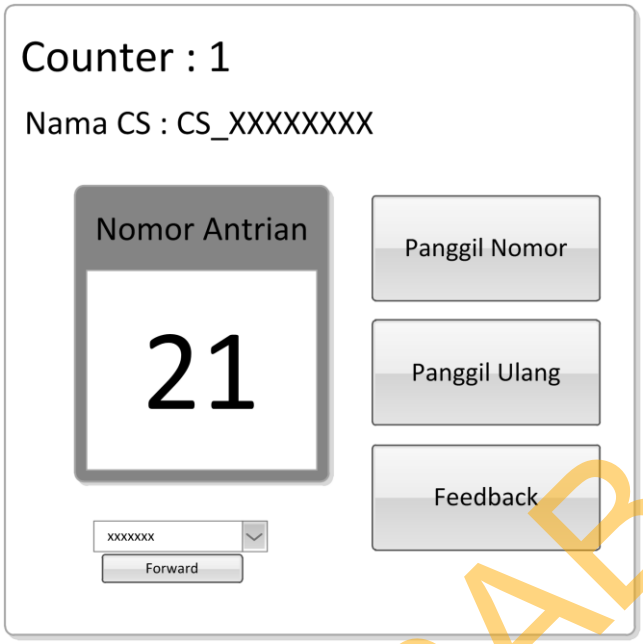
Gambar 3.22 form *login user* pada *form* antrianClient

Gambar 3.22 adalah *form login* untuk *form* antrianClient, sebagai hak akses masuk ke aplikasi.



Gambar 3.23 *form* untuk mengatur *id counter*

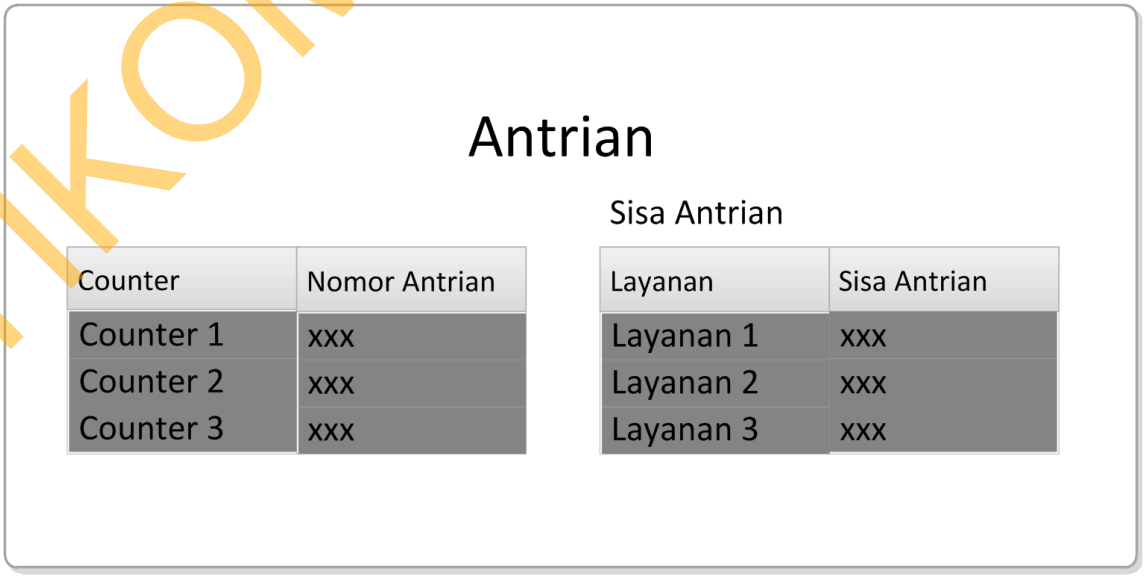
Gambar 3.23 *form* untuk mengatur *id counter*. Dimana *form* ini hanya bisa diakses oleh admin yang *login* di *form* antrianClient.



Gambar 3.24 *form* antrianClient

Gambar 3.28 *form* untuk memanggil nomor antrian. Menampilkan *id counter*, nama CS serta nomor yang dipanggil.

3.9.4. Display Nomor Antrian



Gambar 3.25 *form Display* Nomor Antrian

Gambar 3.25 *form Display* Nomor Antrian berfungsi menampilkan nomor antrian yang dipanggil oleh *customer service* dan *counter* mana yang akan melayani nomor antrian bersangkutan.